PROYECTO FINAL DE CARRERA

TÍTULO DEL PFC: Sistema de gestión de productos con emulación de RFID mediante sensores IEEE 802.15.4

TITULACIÓN: Ingeniería de Telecomunicaciones (segundo ciclo)

AUTOR: Eloy Fernández López

DIRECTOR: Carles Gómez Montenegro

FECHA: 21 de octubre de 2008

Título: Sistema de gestión de productos con emulación de RFID mediante sensores IEEE 802.15.4

Autor: Eloy Fernández López

Director: Carles Gómez Montenegro

Fecha: 21 de octubre de 2008

Resumen

Desde hace unos años, está en auge una tecnología de identificación de productos, llamada RFID, con mayores prestaciones que la tecnología basada en el uso de código de barras. A fecha de hoy, la tecnología RFID ya está madura y está empezando a aumentar su comercialización y uso generalizado.

Este proyecto contiene el diseño, funcionamiento y validación de dos aplicaciones para su uso tanto en almacenes como en comercios.

- La primera de ellas, “aplicacionPC”, es la encargada de procesar toda la información enviada por los motes TelosB que se encuentran adheridos a los productos y los cuales poseen la información referente a ese producto. Una vez procesada, esta información es almacenada en una base de datos donde se lleva el control de todos los productos en stock del almacén. Además cada 15 minutos realiza una actualización automática de dicha tabla, para llevar un control actualizado de los productos que se encuentran en el almacén.

- La segunda de ellas, “aplicacionMovil”, es una aplicación que va destinada principalmente a encargados, jefes y comerciales, para que éstos mediante una pequeña aplicación que tienen que instalar en sus respectivos móviles puedan realizar consultas sobre los productos en stock que se disponen en el almacén.

Finalmente, se han realizado pruebas con las dos aplicaciones que han permitido validar el correcto funcionamiento y eficiencia de ambas. De las pruebas ejecutadas se han obtenido resultados muy próximos a las expectativas previstas, destacando su fiabilidad.

Title: System of product management with RFID emulation through IEEE 802.15.4 sensors

Author: Eloy Fernández López

Director: Carles Gómez Montenegro

Date: October, 21th 2008

Overview

From a few years now, it is in rise a technology of products identification called RFID, that has more features than the technology based on the use of bar codes. At the present time, RFID technology has thoughned up and both its commercialisation and use have increased.

This project contains the design, functioning and validation of two applications for its use both in stores and trades or shops.

- The first one, “aplicacionPC”, is in charge of processing all the information sent by the motes TelosB that are adhered to the products which posses the information related to the product. Once processed, this information is stored in a database where all the products in the stock af the store are under control. In addition, every 15 minutes an automatic update of the table of the products is realised to take a real time control of the products that are in the store.
- The second one, “aplicacionMovil”, is an application that is intended, mainly, to managers, chiefs and commercial advisors of any company, so they can know at any time, the products they have stored in stock, through an application that they should install in their mobile phones.

To end, we should say that tests have been realised to check these applications, and these tests have permited the validation of the correct functioning and efficiency of both applications. Also, from the tests there have been obtained results very close to the expected and we should emphasise the reliability of the applications.

ÍNDICE

INTRODUCCIÓN

Motivación

La tecnología RFID promete revolucionar la vida de las personas. Día a día, aparecen en el mercado dispositivos RFID con mayores capacidades (memoria) y con costes cada vez más bajos. Los estándares se van consolidando, lo que da como resultado que, en un futuro muy próximo, estos dispositivos estén por todas partes. Por ello se debe estar preparado para poder explotar todas estas funcionalidades que nos pueden llegar a proporcionar.

Referente al interés personal por la elección de realizar aplicaciones para ser utilizadas con la tecnología RFID vienen motivas por:

- Es una tecnología que se encuentra en pleno auge y que en un periodo de tiempo no muy grande sustituirá en gran medida al actual sistema de código de barras para la identificación de productos.

- Y, sobretodo, de las oportunidades que la tecnología RFID ofrece al ingeniero de telecomunicaciones para diseñar herramientas y sistemas que reducen drásticamente los costes y el tiempo necesario para recoger, almacenar, procesar, transmitir y analizar de forma automatizada la información de sensado facilitada por redes sensoras.

Objetivos

El objetivo de este proyecto es el de realizar una herramienta para facilitar el control en tiempo real de todos los productos de un almacén.

Para ello, primero se ha realizado una aplicación capaz de recibir y procesar todos los mensajes enviados por los motes adheridos a los productos y con la información de éstos, y guardar toda esta información en una base de datos, la cual podrá ser consultada por los usuarios que tengan acceso a la aplicación.

La segunda aplicación va destinada a su uso en dispositivos móviles, desde los cuales mediante una petición por parte del usuario, podrá saber en todo momento y desde cualquier punto geográfico los productos es stock que hay en el almacén.

Estructura

El proyecto está estructurado en seis capítulos. En el capítulo inicial se presenta una breve introducción sobre los fundamentos de la tecnología RFID, así como su historia y su funcionamiento. El segundo capítulo describe las tecnologías utilizadas para el desarrollo de este proyecto. En el tercer capítulo

se detallan los requisitos que debe cumplir el sistema, es decir, los requisitos de las aplicaciones, para su correcto funcionamiento. El cuarto capítulo muestra el diseño y la implementación tanto de las aplicaciones realizadas como de otros sistemas necesarios como la base de datos. En el quinto capítulo se explican las diferentes pruebas realizadas para la comprobación del correcto funcionamiento de las aplicaciones desarrolladas. Ya por último en el sexto capítulo se detallan las conclusiones de este PFC, así como las líneas futuras que se proponen a partir de él. Finalmente, se comentan las implicaciones medioambientales del PFC.

Aparte de los capítulos mencionados, esta memoria incluye una serie de anexos que los completan y que serán referenciados en los capítulos que corresponda.

CAPÍTULO 1. TECNOLOGÍA RFID

En este primer capítulo se describen los aspectos más importantes de la tecnología RFID, que cada vez más se está asentando como método de identificación, así como su historia, funcionamiento, tipos de dispositivos y estándares.

1.1. Definición

RFID, que responde a las siglas de *Radio Frequency IDentification*, en español Identificación por radiofrecuencia, es un método de almacenamiento y recuperación de datos remoto que usa unos dispositivos llamados etiquetas, transpondedores o *tags* RFID (véase Fig. 1.1). Estas etiquetas son similares a una pegatina, que pueden ser adheridas a un producto, animal o persona. Las etiquetas RFID incorporan antenas para permitir recibir y responder a peticiones por radiofrecuencia desde un emisor-receptor RFID.

El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio. Las tecnologías RFID se agrupan dentro de las denominadas Auto ID (*Automatic Identification*, o Identificación Automática) [1] [2].

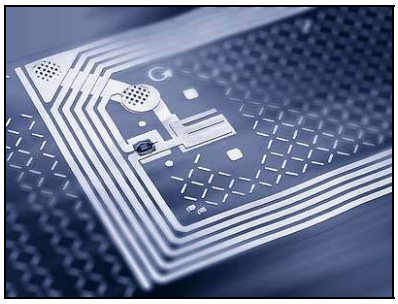

Fig. 1.1 Ejemplo de una etiqueta RFID

1.2. Historia

El origen del RFID está tristemente relacionado con la guerra, concretamente con la II Guerra Mundial, en la que el uso del radar permitía la detección de aviones a kilómetros de distancia, pero no su identificación. El ejército alemán descubrió que si los pilotos balanceaban sus aviones al volver a la base cambiaría la señal de radio reflejada de vuelta. Este método hacía así distinguir a los aviones alemanes de los aliados y se convirtió en el primer dispositivo de RFID pasivo.

Los sistemas de radar y de comunicaciones por radiofrecuencia avanzaron en las décadas de los 50 y los 60 para tratar de identificar objetos remotamente. Las compañías pronto comenzaron a trabajar con sistemas antirrobo que usando ondas de radio determinaban si un objeto había sido pagado o no a la salida de las tiendas.

Las primeras patentes para dispositivos RFID fueron solicitadas en Estados Unidos, concretamente en enero de 1973 cuando Mario W. Cardullo se presentó con una etiqueta RFID activa, la cual disponía de una memoria rescribible. El mismo año, Charles Walton recibió la patente para un sistema RFID pasivo que abría las puertas sin necesidad de llaves. Una tarjeta con un transponedor comunicaba una señal al lector de la puerta que cuando validaba la tarjeta desbloqueaba la cerradura [3].

Después han ido llegando mejoras, tanto en la miniaturización y automatización de los procesos de fabricación, lo cual a llevado a una reducción del precio final, como en la capacidad de emisión y recepción, y en la distancia entre los dispositivos emisor y receptor, consiguiendo extender su uso en ámbitos tanto domésticos como de seguridad nacional, como sucede con el pasaporte expedido en la actualidad en los EEUU que lleva asociadas etiquetas RFID.

1.2.1. El código de barras, antecesor de la tecnología RFID

El código de barras es la tecnología más conocida y extendida para la identificación de productos. Este código se basa en la representación mediante un conjunto de líneas paralelas verticales de distinto grosor y espaciado que en su conjunto contienen una determinada información [4].

Los códigos de barras, que se usan actualmente, llevan más de 35 años en funcionamiento, pero el auge de las nuevas tecnologías pone al descubierto las muchas limitaciones de éste. A continuación se enumeran las diferentes ventajas de las etiquetas RFID [5]:

- A diferencia del código de barras, las etiquetas electrónicas no necesitan contacto visual con el módulo lector para que éste pueda leerlas. La lectura se puede realizar a una distancia de hasta 10 metros.

- Mientras el código de barras identifica un tipo de producto, las etiquetas electrónicas identifican cada producto individual. Es decir, dos productos iguales llevan ahora el mismo código de barras y, por lo tanto, la misma identificación, pero si estuvieran equipados con etiquetas electrónicas se podrían identificar y gestionar de forma individual.

- La tecnología RFID permite leer múltiples etiquetas electrónicas simultáneamente. Los códigos de barras, por lo contrario, tienen que ser leídos secuencialmente. Esta característica del sistema de autoidentificación por radiofrecuencia ofrece diversas ventajas como, por

ejemplo, la reducción del tiempo de espera en las colas de los supermercados.

- Las etiquetas electrónicas pueden almacenar mucha más información sobre un producto que el código de barras, que sólo puede contener un código.

- Mientras que sobre el código de barras se puede escribir sólo una vez, existen etiquetas electrónicas sobre las que se pueden escribir más de una vez.

- La tecnología RFID evita falsificaciones. Con una simple fotocopia se puede reproducir un código de barras. Las etiquetas electrónicas, en cambio, no se pueden copiar.

- Un código de barras se estropea o se rompe fácilmente, mientras que una etiqueta electrónica es más resistente porque, normalmente, forma parte del producto o se coloca bajo una superficie protectora y soporta mejor la humedad y la temperatura.

En resumen, la tecnología RFID resuelve casi todas las limitaciones del código de barras. El único y principal inconveniente de éstas con respecto a los códigos de barra, a fecha de hoy, es su mayor coste, aunque éste se encuentra en un continuo descenso.

1.3. Funcionamiento

El modo de funcionamiento de los sistemas RFID es simple. La etiqueta RFID, que contiene los datos de identificación del objeto al que se encuentra adherido, genera una señal de radiofrecuencia con dichos datos. Esta señal puede ser captada por un lector RFID, el cual se encarga de leer la información y transferirla, en formato digital, a la aplicación específica que utiliza RFID.

Por tanto, un sistema RFID consta de los siguientes tres componentes:

- Etiqueta RFID: compuesta por una antena, un transductor radio y un material encapsulado o chip.

- Lector RFID: compuesto por una antena, un transceptor y un decodificador.

- Subsistema de procesamiento de datos: proporciona los medios de proceso y almacenamiento de datos.

En la Fig. 1.2 se muestra el esquema completo del principio de funcionamiento.

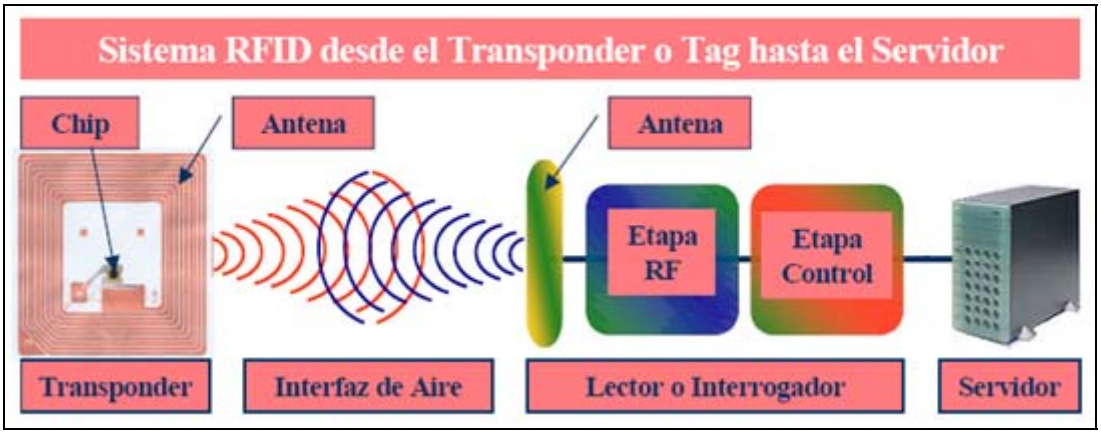

Fig. 1.2 Principio de funcionamiento de un sistema RFID

1.4. Dispositivos

En este punto se describirán los dos elementos básicos de un sistema RFID: los lectores y las etiquetas. De éstas últimas se pueden encontrar de tres tipos diferentes: activas, semipasivas (o semiactivas, también conocidas como asistidas por batería) o pasivas.

1.4.1. Lector RFID

El lector RFID es un dispositivo que emite señales de radio a una frecuencia predeterminada, con el fin de interrogar a la etiqueta RFID y obtener una respuesta. Cuando obtiene esta respuesta, convierte la señal de radiofrecuencia en un código numérico que puede ser transmitido a otros dispositivos o sistemas. Los lectores RFID pueden variar en tamaño, funcionalidad y coste, éste último fluctuando desde unos cuantos cientos de euros hasta miles de euros, dependiendo de su nivel de complejidad y sofisticación. En la Fig. 1.3 se muestra un lector RFID móvil.

Fig. 1.3 Lector RFID móvil

1.4.2. Tipos de etiquetas RFID: pasivas, semipasivas y activas

Las etiquetas **RFID pasivas** no contienen fuente de alimentación propia, obtienen la corriente eléctrica de la señal de radiofrecuencia del lector cuando está escaneando dicha etiqueta. Gracias a esta energía, las etiquetas son capaces de transmitir una respuesta al lector.

Las etiquetas pasivas, en la práctica tienen distancias de lectura que varían entre unos 10 milímetros hasta cerca de 6 metros dependiendo del tamaño de la antena de la etiqueta, de la potencia y de la frecuencia en la que opera el lector. La Fig. 1.4 muestra una etiqueta RFID pasiva.

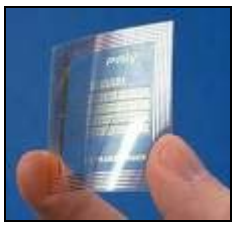

Fig. 1.4 Etiqueta RFID pasiva

Las etiquetas **RFID semipasivas** son muy similares a las pasivas, salvo que incorporan además una pequeña batería. Esta batería permite a la etiqueta estar constantemente alimentada, eliminando la necesidad de diseñar una antena para recoger potencia de una señal entrante. Las etiquetas RFID semipasivas responden más rápidamente comparadas con las etiquetas pasivas. En la Fig. 1.5 se muestran las partes que forman una etiqueta semipasiva.

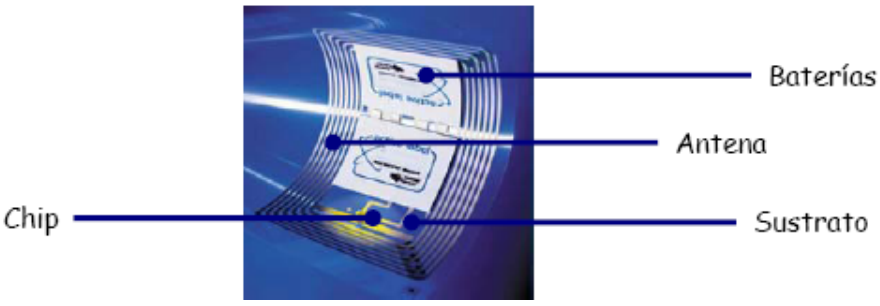

Fig. 1.5 Etiqueta RFID semipasiva

Finalmente, las etiquetas **RFID activas**, deben tener una fuente de energía, y pueden tener rangos mayores de cobertura y memorias más grandes que las etiquetas pasivas, así como la capacidad de poder almacenar información adicional enviada por el transmisor-receptor.

Actualmente, las etiquetas activas más pequeñas tienen un tamaño aproximado de una moneda. Muchas etiquetas activas tienen rangos de cobertura de diez metros, y una duración de batería de hasta varios años. En la Fig. 1.6 se muestra una etiqueta RFID activa con un encapsulado en forma de llavero.

Fig. 1.6 Etiqueta RFID activa con encapsulado en forma de llavero

1.5. Estandarización y regulación

Los estándares para RFID son un tema delicado ya que muchas aplicaciones en las que se involucran estas etiquetas inteligentes están relacionadas con los pagos o con las cadenas de producción. Los estándares para RFID tratan los siguientes temas:

- Protocolo en el interfaz aire: la forma en la que las etiquetas y los lectores se pueden comunicar.
- Contenido de los datos: organización de los datos a intercambiar.
- Conformidad: pruebas que los productos deben cumplir para estar dentro del estándar.
- Aplicaciones: cómo se pueden utilizar las aplicaciones con RFID.

La generación de estándares para RFID tiene como particularidad el conflicto entre dos organizaciones: ISO y EPC Global, con propuestas de estándares por ambas partes y ambas buscan conseguir etiquetas de bajo coste.

Los estándares de la EPC para etiquetas son de dos clases:

- Clase 1: Etiqueta simple, pasiva, de sólo lectura con una memoria no volátil programable una sola vez.
- Clase 2: Etiqueta de sólo lectura que se programa en el momento de fabricación del chip (no programable a posteriori).

Por su parte, ISO ha desarrollado estándares de RFID para la identificación automática y la gestión de objetos. Existen varios estándares relacionados, como ISO 10536, ISO 14443 y ISO 15693, pero la serie de estándares estrictamente relacionada con las RFID y las frecuencias empleadas en dichos sistemas, es la serie 18000.

Por otra parte respecto la regulación de las frecuencias, no hay ninguna corporación pública global que gobierne las frecuencias usadas para RFID. En principio, cada país puede fijar sus propias reglas. Las principales corporaciones que gobiernan la asignación de las frecuencias para RFID son:

- EEUU: FCC (*Federal Communications Commission*)
- Canadá: DOC (Departamento de la Comunicación)
- Europa: ERO, CEPT, ETSI y administraciones nacionales. Obsérvese que las administraciones nacionales tienen que ratificar el uso de una frecuencia específica antes de que pueda ser utilizada en ese país.
- Japón: MPHPT (*Ministry of Public Management, Home Affairs, Post and Telecommunication*)
- China: Ministerio de la Industria de Información
- Australia: Autoridad Australiana de la Comunicación (*Australian Communication Authority*)
- Nueva Zelanda: Ministerio de desarrollo económico de Nueva Zelanda (*New Zealand Ministry of Economic Development*

Las etiquetas RFID de LF (125 - 134 kHz y 140 - 148.5 kHz) y de HF (13.56 MHz) se pueden utilizar de forma global sin necesidad de licencia.

La banda de UHF (868 - 928 MHz) no puede ser utilizada de forma global, ya que no hay un único estándar global. En Norteamérica, UHF se puede utilizar sin licencia para frecuencias entre 908 y 928 MHz, pero hay restricciones referentes a la energía de transmisión. En Europa, UHF está bajo consideración para frecuencias entre 865.6 y 867.6 MHz. Su uso es sin licencia sólo para el rango de 869.40 y 869.65 MHz, pero también tiene restricciones en la potencia de transmisión. El estándar UHF norteamericano (908 - 928 MHz) no es aceptado en Francia ya que interfiere con sus bandas militares.

En China y Japón no hay regulación para el uso de UHF. Cada aplicación de esta frecuencia en estos países necesita de una licencia, que debe ser solicitada a las autoridades locales, y puede ser revocada. En Australia y Nueva Zelanda, el rango es de 918 a 926 MHz para uso sin licencia, pero, como en los demás países, hay restricciones en la energía de transmisión.

CAPÍTULO 2. Tecnologías utilizadas

En este capítulo se van a explicar las diferentes tecnologías utilizadas para el desarrollo de este PFC, como son la plataforma utilizada por los motes TelosB, la plataforma Java2 utilizada para el desarrollo de las aplicaciones, y el servidor y la base de datos.

2.1. Plataforma TelosB

2.1.1. Entorno de desarrollo

TelosB es una plataforma experimental desarrollada y distribuida por la *University of California at Berkeley* y *Crossbow Technology Inc*. En la Fig. 2.1 se puede ver la imagen de un dispositivo TelosB.

Fig. 2.1 Fotografía de un mote TelosB

Se han utilizado este tipo de sensores, ya que se trata del material disponible para este PFC, además dispone de una plataforma ya conocida y emula perfectamente a una etiqueta RFID, pero para una futura implementación comercial se han de utilizar las etiquetas RFID pasivas explicadas en el Capítulo 1 [6].

Otra ventaja muy importante de este tipo de sensores es que poseen una gran facilidad de programación y recolección de datos por parte del usuario, ya que disponen de una interfaz USB.

Sus características técnicas más importantes son las siguientes:

- Soporta el interfaz radio IEEE 802.15.4, con velocidad de transmisión de 250 kbps, véase apartado 2.1.2.
- Microcontrolador MSP430 de 8 MHz con 10 kB RAM.
- Antena integrada, banda de 2.4 a 2.4835 GHz.
- Recolector de datos de programación vía USB.

- Sistema operativo de libre distribución (funciona con TinyOS 1.1.11 o superior).
- Bajo consumo de potencia.
- Funciona con dos pilas estándar AA.

En los puntos 2.1.3 y 2.1.4 se realizará una pequeña descripción del sistema operativo TinyOS y del lenguaje de programación que éste utiliza, denominado nesC.

2.1.2. Estándar IEEE 802.15.4 (ZigBee)

2.1.2.1. Introducción

El estándar IEEE 802.15.4 [7] fue creado para cubrir la necesidad del mercado de estándares inalámbricos de baja tasa para aplicaciones en redes de sensores. Los estándares existentes hasta el momento en el mercado estaban destinados a aplicaciones con mayores requisitos en cuanto a ancho de banda se refiere, como pueden ser videoconferencias o redes domésticas. Los ejemplos más representativos de estas tendencias son el IEEE 802.11, también conocido como WIFI, y el IEEE 802.15.1, Bluetooth. Los inconvenientes que surgían al utilizar cualquiera de éstos, eran su gran consumo de energía y ancho de banda frente a la baja tasa y bajos requisitos de energía necesaria para las redes de sensores.

La necesidad de este nuevo estándar impulsó la creación de distintos grupos de trabajo, como por ejemplo, el IEEE 802.15 WPAN Task Group 4 [8], con el objetivo de investigar en comunicaciones con baja tasa de transmisión y el cual publicó en octubre de 2003 las especificaciones para el estándar IEEE 802.15.4. Seis meses después una agrupación de compañías para el desarrollo de aplicaciones sobre este tipo de redes, llamada ZigBee Alliance [9] publicó la especificación ZigBee v1.0 indicando recomendaciones para la creación de aplicaciones funcionando sobre dicho estándar.

De esta manera, el estándar 802.15.4 define la capa física y la subcapa MAC para las redes LR-WPAN, mientras que ZigBee Alliance, aprovechando y basándose en las capas inferiores definidas en el estándar 802.15.4, trabaja en las capas superiores y define la capa de aplicación, la capa de red y los servicios de seguridad.

IEEE 802.15.4 es una tecnología inalámbrica que nos permite comunicaciones de corto alcance (distancias hasta unos 75 metros), y bajo consumo. Puede funcionar en la banda de 2.4 GHz a una tasa de 250 Kb/s, en la de 868 MHz a 40 Kb/s y en la de 915 MHz a 20 Kb/s, aunque la mayoría de fabricantes optarán por la elección de la primera ya que puede ser usada en todo el mundo, mientras que las dos últimas sólo se pueden usar en Europa y EEUU, respectivamente.

El objetivo es que un sensor equipado con esta tecnología pueda ser alimentado con dos pilas AA un periodo de entre seis meses y dos años, aunque en la práctica se ha verificado que se podrán conseguir casi cinco años de duración en las aplicaciones de domótica y seguridad.

En conclusión, IEEE 802.15.4 resulta adecuado para redes estáticas, escalables, con muchos dispositivos, pocos requisitos de ancho de banda y dónde se requiera una duración muy prolongada de la batería, como pueden ser aplicaciones de control o monitorización, en multitud de entornos.

En el siguiente apartado se explicará el funcionamiento del estándar IEEE 802.15.4.

2.1.2.2. Funcionamiento

En el estándar IEEE 802.15.4 se indican dos modos de funcionamiento, el modo *beaconless* y el modo *beacon*. A continuación se describirá de forma breve las dos formas existentes.

La transmisión en modo *beacon* está orientada a redes donde existe el papel del coordinador (por ejemplo en el caso de una topología en estrella), el *Personal Area Network coordinator* se encarga de transmitir *beacons* cada cierto tiempo para que los dispositivos dentro de su red se puedan sincronizar, definiendo en caso de necesidad una latencia máxima para aquellos dispositivos que necesiten tener este parámetro garantizado (transmisión con *beacons* y un tiempo de acceso garantizado). Como mecanismo de acceso al medio se utiliza CSMA-CA ranurado.

La transmisión en modo *beaconless* está orientada a redes *peer to peer* donde existe comunicación entre todos sin la intervención de un coordinador que los sincronice. El mecanismo de acceso al medio es el CSMA-CA no ranurado de modo que cada dispositivo transmite en el momento que es necesario sin esperar ningún *beacon* de un *PAN coordinator*.

A continuación vamos a profundizar un poco más en este tipo de funcionamiento ya que es el que implementa el chip de los sensores utilizados en este proyecto (chip CC2420). En el Anexo C se muestra el archivo de configuración de este chip y se definen las variables más importantes, como el canal o la potencia que utiliza para trabajar.

2.1.2.3. Formato del mensaje AM para la MAC del chip CC2420

El formato del mensaje radio que se observa al recuperar cualquier paquete se compone de dos campos: un primer campo, el campo direccionamiento, dentro del cual cada mote coloca su cabecera definida en el archivo de configuración AM.h (véase Anexo C), y un segundo campo, el campo de datos, con un tamaño máximo de 102 bytes.

En la Fig. 2.2 se muestra el formato de la trama.

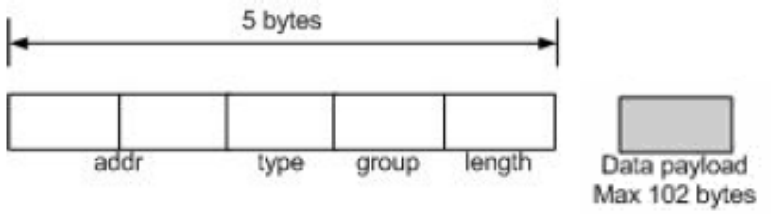

Fig. 2.2 Formato de la trama del paquete radio de los motes

Como se verá en el apartado 4.2, para nuestro proyecto sólo nos interesa el campo *data*, que es donde se encuentra la información referente al producto, es decir, el código de la marca del producto, el código de producto y el código único que identifica a ese producto.

2.1.3. TinyOS

El sistema operativo TinyOS es de libre distribución y está especialmente diseñado para dispositivos sensores. Es un sistema operativo basado en componentes, tareas y orientado a eventos, todo ello con el uso de poca memoria.

La dinámica de los programas realizados bajo TinyOS está conducida por tareas, las cuales pueden ser interrumpidas para atender eventos ya que éstos disponen de una prioridad mayor para ser atendidos.

TinyOS define 3 clases de abstracción que son la esencia para su comprensión y posterior programación:

- Los comandos se pueden observar en la Fig. 2.3. Al llamar a un comando, éste dentro de su componente llama a otros componentes de capas inferiores.

- Los eventos conllevan el efecto inverso, una llamada hacia arriba como se puede ver en la Fig. 2.3. Un componente de bajo nivel avisa a uno de alto nivel de que ha sucedido algo. Si sucede un evento, éste tiene mayor prioridad que cualquier tarea y pasaría a ejecutarse.

- Las tareas son porciones de código que se ejecutan de forma asíncrona siempre y cuando la CPU no tenga que ejecutar ningún evento. Éstas se ejecutan por orden de llamada (FIFO) y en caso de que la cola estuviese vacía el procesador entraría en *standby* hasta que un nuevo evento hiciese despertar al procesador.

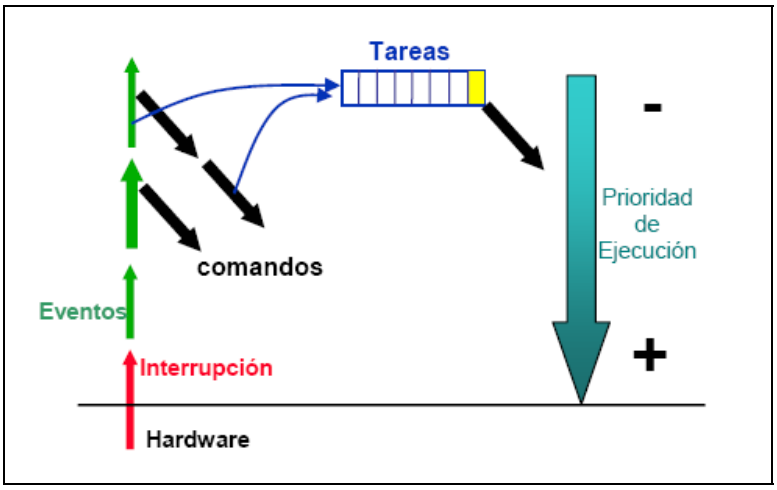

Fig. 2.3 Esquema de funcionamiento de TinyOS

Para programar mediante TinyOS se utiliza una extensión del lenguaje C llamada nesC, la cual permite el uso de interfaces que van conectadas a componentes.

Para crear un programa en nesC, únicamente se tiene que elegir los componentes (las partes funcionales) previamente programadas y después relacionarlas mediante otro fichero de configuración a las interfaces que serán utilizadas como código en nuestra futura aplicación.

2.1.4. nesC

nesC es un lenguaje de programación que se utiliza para crear aplicaciones que serán ejecutadas en sensores con el sistema operativo TinyOS. Por tanto, dicho lenguaje, proporciona características necesarias para poder realizar aplicaciones de una forma más cómoda para el programador.

La característica principal de este lenguaje de programación es que combina las ventajas de una programación orientada en objetos (POO) y de una programación orientada en eventos (POE). La POO permite programar los diferentes componentes de los sensores de forma genérica.

En nesC se definen dos clases de componentes: los módulos y las configuraciones. Los módulos contienen el código en sí del componente mediante interfaces. Por otra parte, las configuraciones enlazan o relacionan las interfaces de los componentes que se van a necesitar en nuestro módulo. El fichero de configuración es crucial porque si no está bien configurado, la aplicación no compilará. Tanto módulos como configuraciones utilizan la misma extensión *.nc. a diferencia entre ambos está en la sintaxis que utilizan y es por ello que siguen un código de letras para diferenciar unos de otros. Por ejemplo, cuando trabajamos con un programa “aplicación”, aplicacionM.nc sería el módulo y aplicacion.nc seria el fichero de configuración.

2.2. J2EE (Java 2 Platform, Enterprise Edition)

Se ha tomado la decisión de utilizar la tecnología J2EE como herramienta de trabajo para la realización de la parte aplicación del proyecto ya que se trata de una tecnología libre (*Open Source*) de la cual existen multitud de herramientas gratuitas. Pero no sólo esto, sino que existe una cantidad de documentación y APIs superior a la que pueda existir para otras plataformas como .NET y ésto es un factor muy importante ya que puede facilitar y agilizar enormemente el diseño e implementación de la aplicación. Otro aspecto importante, es que uno de los objetivos del proyecto es fomentar el uso de las tecnologías libres a nivel empresarial y poder demostrar que se pueden realizar grandes proyectos sin necesidad de tener que invertir en herramientas que en muchos casos pueden resultar extremadamente costosas. A continuación se muestra el listado completo de motivos por los cuales se ha elegido J2EE como tecnología de desarrollo:

- Lenguaje JAVA extendido
- Tecnología potente y madura
- Interoperable con otras tecnologías: XML, JavaScript, HTML, …
- Gran cantidad de soporte en la red: APIs, manuales…
- *OpenSource* y herramientas de desarrollo gratuitas (Eclipse)
- Muchas utilidades ya creadas y fáciles de integrar
- Fácil conectividad con Base de Datos: *driver* JDBC (MySQL)
- Existencia de *frameworks* de desarrollo basados en MVC[1] (Hibernate)

2.2.1. Frameworks para J2EE

En el desarrollo de software, un *framework* es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado. Típicamente, un *framework* puede incluir soporte de programas, bibliotecas y un lenguaje de *scripting* para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Provee de una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

En general, el término *framework*, se está refiriendo a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un *framework* se puede considerar como una aplicación genérica incompleta y configurable a la que se le puede añadir las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un *framework* son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. Para facilitar el desarrollo de las aplicaciones J2EE se han ideado varios *frameworks* de los cuales se destacan los mostrados en la Tabla 2.1.

[1] Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control.

Tabla 2.1. *Framework* y su función

FRAMEWORK	FUNCIÓN
Java Server Faces	Creación de interfaces
Hibernate	Facilitar la conectividad con BBDD
Spring	Facilitar la configuración mediante beans
Struts	Facilitar el control de eventos

- ***Java Server Faces:*** orientado a la creación de interfaces de usuario.

- ***Hibernate:*** tiene como objetivo facilitar la persistencia de objetos Java en bases de datos y al mismo tiempo la consulta de estas bases de datos para obtener objetos. Éste será muy útil en el desarrollo de nuestras aplicaciones.

- ***Spring:*** tiene como objetivo facilitar la configuración de los java *beans*[2] dentro de una aplicación. Su meta es conseguir separar los accesos a datos y los aspectos relacionados con las transacciones, para permitir objetos de la capa de negocio[3] reutilizables que no dependan de ninguna estrategia de acceso a datos o transacciones.

- ***Struts:*** orientado al control de eventos.

2.3. J2ME (Java 2 Platform, Micro Edition)

Aunque existen varias plataformas para el desarrollo de aplicaciones para dispositivos móviles, como pueden ser Microsoft .net Compact Framework o J2ME, se ha decidido utilizar esta última ya que Microsoft .net tiene dos desventajas muy importantes sobre J2ME. La primera, es que no es multiplataforma, no pudiendo utilizarse bajo ningún otro sistema operativo que no sea de Microsoft y la segunda es que no es gratuita.

2.3.1. J2ME

J2ME es la versión de Java orientada a los dispositivos móviles. Debido a que los dispositivos móviles tienen una potencia de cálculo baja e interfaces de usuario pobres, es necesaria una versión específica de Java destinada a estos dispositivos, ya que el resto de versiones de Java, J2SE[4] o J2EE, no encajan dentro de este esquema. J2ME es por tanto, una versión "reducida" de J2SE.

[2] Un bean es un componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno (p.ej. un botón).
[3] La capa de servicios de negocio consiste en la lógica que realiza las funciones principales de la aplicación: procesamiento de datos, coordinación de varios usuarios o administración de recursos externos como, por ejemplo, bases de datos.
[4] J2SE es la versión de Java orientada a aplicaciones de escritorio y que no desarrollen WebServices.

2.3.2. Plataforma

En la Fig. 2.4 se muestra qué lugar ocupa la plataforma J2ME en el mundo Java. Dicha plataforma se divide en configuraciones y en perfiles por encima de ellas [10].

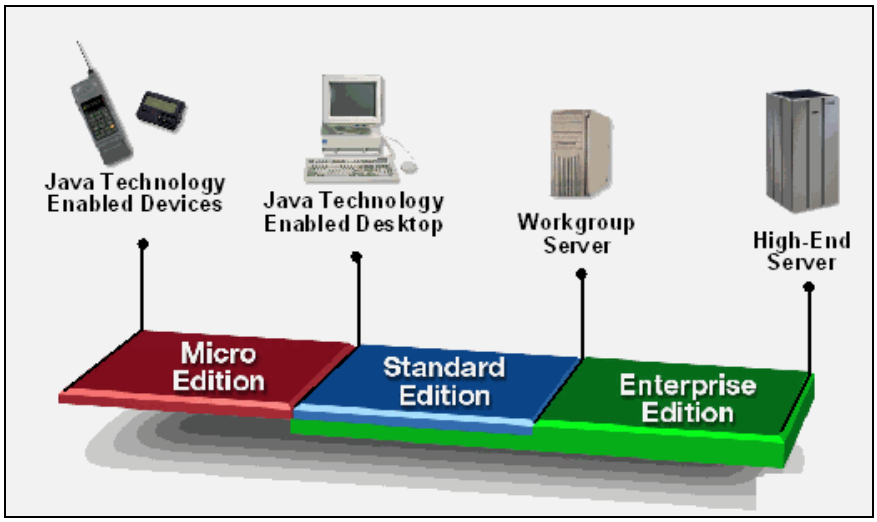

Fig. 2.4 Plataforma Java2

En función de las características del dispositivo para el cual se desarrolla la aplicación, habrá que acogerse a las normas impuestas por alguna de las especificaciones que aparecen en el gráfico. En este caso, al orientarnos al desarrollo para dispositivos móviles de capacidades muy restringidas en gráfica, procesamiento y memoria, se trabajará bajo la combinación siguiente:

- Configuración *Connected Limited Device Configuration*, CLDC (versión 1.1), a la cual nos obliga la reducida *Kilo Virtual Machine* (KVM) que utilizan estos dispositivos para interpretar los *bytecodes*[5] Java que generemos.

- Perfil *Mobile Information Device Profile,* MIDP, en su última y mejorada versión 2.0.

2.3.3. Aplicaciones J2ME. MIDlet

Una aplicación Java que cumpla las especificaciones CLDC y MIDP será denominada MIDlet, y a varias de ellas empaquetadas en un mismo elemento (JAR) se le denominará MIDlet SUITE. El dispositivo nos permitirá seleccionar una u otra aplicación de la *suite* como él considere apropiado.

[5] Código resultado al compilar el código fuente, cercano al lenguaje máquina pero independiente del ordenador y el sistema operativo en que se ejecuta.

Tal y como se muestra en la Fig. 2.5, los MIDlets siempre tendrán tres métodos básicos que marcan los estados de su ciclo de vida [11].

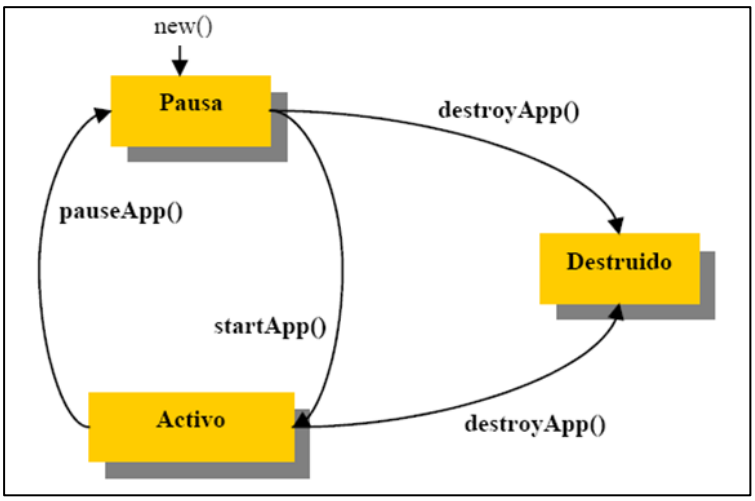

Fig. 2.5 Ciclo de vida de un MIDlet

- **Pausado:** Estado "en espera" en el que el MIDlet mantiene los mínimos recursos posibles, entrando en él cuando se crea (antes de ejecutarse su método startApp()) o tras llamarlo desde el método startApp(). Además, la plataforma puede pasar el MIDlet a este estado si así lo estima oportuno (por ejemplo, ante una llamada telefónica).

- **Activado:** Estado de ejecución del MIDlet al que se pasa tras ejecutar su método startApp(), tanto inicialmente o como después de la recuperación de una pausa.

- **Destruido:** Los dos estados anteriores pueden pasar a éste y de él ya no se podrá salir. Es el estado donde el MIDlet concluye su actividad, pasando a él por medio de la invocación de su método destroyApp() o, por ejemplo, ante alguna excepción que se produzca en el constructor[6] del MIDlet.

2.4. Base de datos

En este punto se van a describir las características de la base de datos. Se ha decidido elegir el gestor de base de datos MySQL ya que se trata de un software muy potente y de libre distribución. MySQL permite elegir entre diferentes tipos de “motores de almacenamiento” que nos permite seleccionar el tipo de almacenamiento interno de cada tabla, en base al que mejor se adecue a nuestra situación.

[6] Un constructor es una función que sirve para construir o inicializar objetos.

Para este proyecto se ha decidido utilizar la tecnología de almacenamiento de tablas InnoDB frente por ejemplo la tecnología MyISAM que es la que viene por defecto en MySQL y la cual ofrece más velocidad de lectura siempre y cuando no ocurran simultáneamente. Las principales características del motor de almacenamiento InnoDB son las siguientes:

- Transaccional[7].
- Multiversionado: cuando múltiples transacciones modifican registros, InnoDB mantiene aisladas las transacciones guardando para cada una de ellas una versión distinta de un mismo registro, a cada transacción la versión que le corresponde.
- Fácil recuperación de datos en caso de error.

En definitiva, es una buena elección cuando se necesitan transacciones, restricciones en claves foráneas, o se tienen muchas escrituras simultáneas, como va a ser el caso de este proyecto, al estar siempre recibiendo los datos enviados por los motes adheridos a los productos [12] [13].

2.4.1. JDBC

JDBC es el acrónimo de *Java Database Connectivity*, un API que permite la ejecución de operaciones sobre base de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el lenguaje SQL del modelo de base de datos que se utilice. En la Fig. 2.6 se puede observar un esquema de la interacción de una aplicación con la base de datos mediante JDBC.

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la librería de conexión apropiada al modelo de su base de datos, es este caso la librería que se utilizará será *mysql-connector-java-5.1.5-bin* y accede a ella estableciendo una conexión. A partir del establecimiento de conexión puede realizar cualquier tipo de tarea con la base de datos siempre que tenga permiso, como pueden ser: consultas, actualizaciones, añadir elementos a las tablas o borrarlos, etc [14].

[7] Un gestor transaccional es un componente que procesa información descomponiéndola de forma unitaria en operaciones indivisibles.

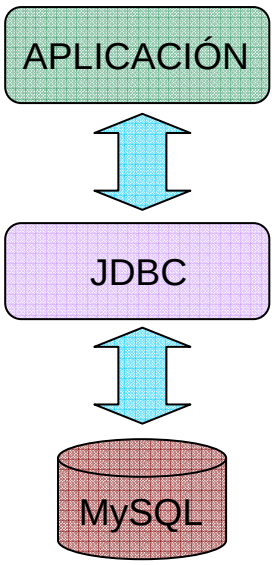

Fig. 2.6 Arquitectura de comunicación base de datos ↔ aplicación

2.5. Servidor Tomcat

Para una parte del proyecto era necesaria la utilización de un servidor que pudiera contener *servlets*[8] y se decidió elegir Tomcat. Tomcat (Jakarta Tomcat) es un servidor de aplicaciones que sirve como contenedor de *servlets* y *Java Server Pages* (JSP) desarrollado bajo el proyecto Jakarta en *Apache Software Foundation*.

El servidor Tomcat puede funcionar como servidor Web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor Web autónomo en entornos con alto nivel de tráfico y alta disponibilidad [15].

La estructura de directorios de Tomcat es importante ya que al instalarlo se genera una variable de entorno CATALINA_HOME que corresponde al *path* donde se encuentra el contenedor de *servlets*. Dicha estructura es [16]:

- bin: contiene los *scripts* de arrancar/parar.
- common: clases comunes que pueden utilizar Catalina[9] y las aplicaciones Web.
- conf: contiene varios ficheros de configuración incluyendo `server.xml` (el fichero de configuración principal de Tomcat) y `web.xml` que configura los valores por defecto para las distintas aplicaciones desplegadas en Tomcat.

[8] Un servlet es un objeto que se ejecuta en un servidor.
[9] Catalina es el nombre del contenedor de servlets del Jakarta Tomcat.

- doc: contiene varia documentación sobre Tomcat.
- lib: contiene varios ficheros .jar que son utilizados por Tomcat.
- logs: donde Tomcat sitúa los ficheros de diario.
- server: clases utilizadas solamente por Catalina.
- shared: clases compartidas por todas las aplicaciones Web.
- webapps: directorio que contiene las aplicaciones Web que cargará el navegador cuando haga la petición al servidor.
- work: almacenamiento temporal de ficheros y directorios.

En la Fig. 2.7 se muestra un sencillo esquema de la posición que ocupa el servidor Tomcat dentro de esta arquitectura.

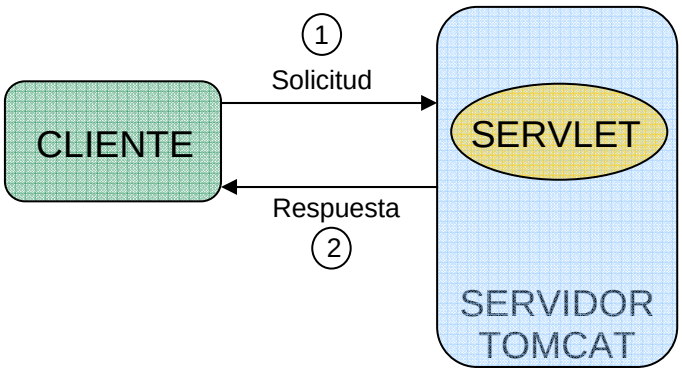

Fig. 2.7 Esquema de arquitectura con servidor Tomcat

CAPÍTULO 3. Requisitos del sistema

El objetivo de este proyecto es el de poder realizar una aplicación que lleve un control en tiempo real de todos los productos de un almacén, y poder acceder a la base de datos de los productos tanto desde el mismo almacén con una aplicación instalada in situ, como desde cualquier otro lugar accediendo mediante una conexión HTTP con una aplicación móvil. Las dos aplicaciones deben ofrecer un entorno amigable y fácilmente ampliable en caso de que el administrador quisiera añadir nuevas funcionalidades en un futuro.

Tal y como se ha explicado en el Capítulo 2, el proyecto se va a realizar utilizando el lenguaje de programación Java en su especificación J2EE para la aplicación instalada en el almacén y J2ME para la aplicación móvil. También se pretende utilizar en la medida que sea posible el máximo de tecnologías de libre distribución.

En este capítulo se presentan los requisitos que debe cumplir el sistema. Se han dividido los requisitos en funcionales y en no funcionales. Los requisitos funcionales son los procesos que se pueden realizar en la plataforma como por ejemplo poder registrar usuarios o productos. Los requisitos no funcionales son condiciones que debe cumplir un proyecto como por ejemplo que la aplicación sea rápida, segura o barata de realizar.

3.1. Requisitos funcionales

Los requisitos funcionales que debe cumplir la aplicación son los siguientes:

- Gestión de administradores
    - Registrar administradores.
    - Modificar el *password* de los administradores.
    - Eliminar administradores.
    - Registrar usuarios.
    - Modificar el *password* de los usuarios.
    - Eliminar usuarios.
    - Registrar marcas de productos.
    - Modificar la descripción de las marcas registradas.
    - Eliminar marcas.
    - Registrar productos de las marcas que se encuentren dadas de alta.
    - Modificar la descripción de los productos registrados.
    - Eliminar productos.

- Gestión de usuarios
    - Añadir productos que por motivos externos a la aplicación no son reconocidos (p.ej. el producto no lleve etiqueta o este dañada).

  - Eliminar productos que por motivos externos a la aplicación no son reconocidos.

- Motor de comunicaciones
  - Servicio de información de los productos en stock mediante conexiones HTTP de la aplicación móvil y el servidor instalado en el almacén.

- Control de acceso:
  - Protocolo de seguridad de acceso de los usuarios mediante *login* y *password* que le restringirá de ciertas acciones.

- Sistema automático: creación de un sistema automático encargado de recoger la información enviada por los motes y que cada cierto tiempo realice una actualización de los productos en stock en la base de datos.

3.2. Requisitos no funcionales

El sistema también tiene que cumplir las siguientes condiciones o requisitos no funcionales:

- Seguridad: el proyecto tiene que garantizar la protección de acceso a ciertas secciones.

- Entorno amigable: el sistema tiene que tener un entorno fácil de usar para que cualquier usuario pueda utilizar los servicios sin dificultades.

- Rápido: el sistema tiene que ser los más eficiente posible en cuanto a velocidad.

- Escalable: el sistema no debería colapsarse al ampliar el número de peticiones por parte de los usuarios móviles, ni por el aumento de productos en stock.

- Coste: el proyecto tiene que conseguir hacerse en el menor número de horas y con uso prioritario de software libre.

- Ampliable: la aplicación tiene que dar facilidades para poderse ampliar el número de secciones o servicios o incluso un cambio de aspecto de la interfaz gráfica.

- Multithread: el sistema puede estar almacenando un producto nuevo y un usuario móvil realizando una petición de forma simultánea.

- Documentación: el proyecto debe estar documentado para permitir futuras mejoras o ampliaciones.

- Control de errores: el sistema debe disponer de un sistema de logs para intentar identificar el mayor número de errores posibles y con rapidez.

3.3. Roles de la plataforma

Se han definido cinco roles o usuarios tipo diferentes que pueden utilizar la aplicación. Se distinguen principalmente por los permisos de que disponen y por las acciones que pueden llegar a desempeñar. A continuación se explica cada uno:

- Usuario público: cualquier persona perteneciente a la empresa puede acceder a la aplicación instalada en el almacén para ver que productos se encuentran en stock, y activar o desactivar la recogida de datos enviados por los motes.

- Usuario registrado: cuando un usuario público se registra pasa a ser usuario registrado y obtiene permisos para realizar ciertas acciones como añadir o eliminar productos en stock mediante su nombre de usuario y su contraseña.

- Usuario móvil: cualquier persona perteneciente a la empresa y con la aplicación instalada en su teléfono móvil, puede realizar peticiones sobre los productos que se encuentran en stock en tiempo real.

- Administrador: será una persona con permisos totales sobre la aplicación. Podrá acceder a la sección desde la cual se añaden, modifican o eliminan administradores, usuarios, marcas y productos.

- Sistema: se trata de un algoritmo inteligente que se encargará de realizar cada cierto tiempo una actualización de los productos en stock en la base de datos.

CAPÍTULO 4. DISEÑO E IMPLEMENTACIÓN

En este capítulo se presenta el diseño e implementación del conjunto del proyecto, tanto la información que se envía por medio radio, como las diferentes aplicaciones realizadas y la información almacenada en la base de datos.

4.1. Diseño global del proyecto

La Fig. 4.1 muestra el escenario con las tres partes principales de las que está formado este proyecto.

La primera de ellas consiste en el código para el envío de la información referente a los productos mediante sus respectivos motes por el medio radio y la recepción de estos mensajes por parte del mote receptor, el cual se encuentra conectado a un puerto USB del PC, donde se encuentra instalada la aplicaciónPC.

La segunda es la aplicaciónPC, la cual es la encargada de procesar los mensajes recibidos por el mote receptor. También ésta es la encargada de administrar los usuarios y los productos e interactuar con la base de datos.

Por último, se tiene la comunicación entre el teléfono móvil donde se encuentra instalada la aplicaciónMóvil y el servidor instalado en el PC donde se ejecuta la aplicación InfoMobile. Esta parte del proyecto permite llevar un control actualizado y desde cualquier sitio de los productos en stock que hay en el almacén, siempre y que se disponga de cobertura y acceso a Internet.

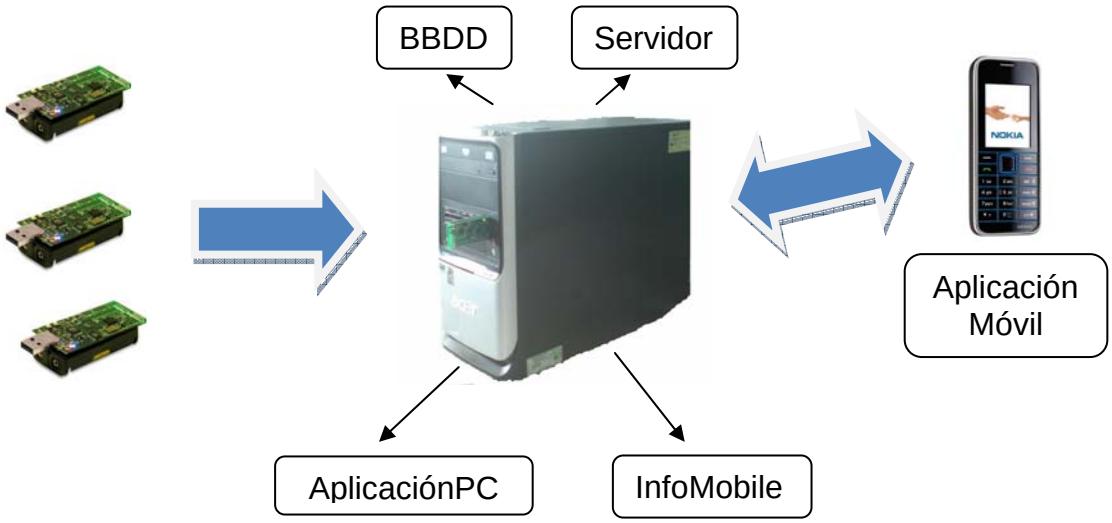

Fig. 4.1 Diseño global del proyecto

4.2. Código ejecutado por los motes TelosB

Como ya se ha dicho en el apartado 2.1.1, los motes utilizados en este proyecto son los TelosB, los cuales tratan de emular un RFID. En la Fig. 4.2 se muestra el formato de la trama del paquete radio que es enviado por éstos y que será recibido por el mote receptor y procesado por la aplicaciónPC.

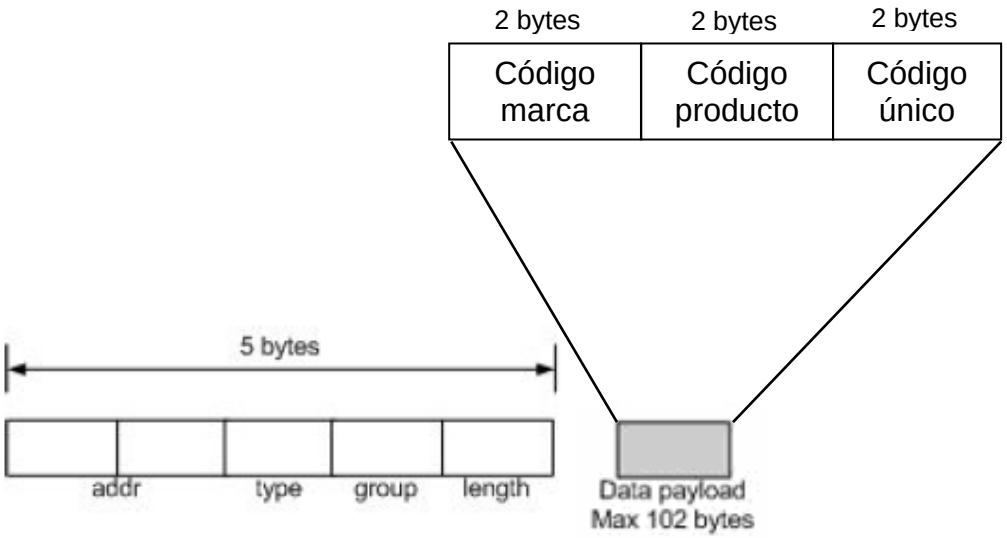

Fig. 4.2 Formato de la trama del paquete radio de los motes TelosB

En este proyecto, del paquete radio enviado por los motes, sólo nos interesan los bytes enviados en el campo *data*. Este campo puede llegar a tener como máximo una longitud de 102 bytes pero para enviar la información referente al producto sólo nos hacen falta 6 bytes, cuya funcionalidad se explicará a continuación. En el campo *length* es donde se define la longitud que tomará el campo *data*, en este caso el campo *lenght* tomará el valor 6 para conseguir la longitud deseada del campo *data*. Para esta aplicación los demás bytes que se envían en el paquete radio no tienen ningún interés.

Para albergar la información referente del producto se han definido 3 campos dentro del campo *data*, de 2 bytes cada uno, por lo tanto cada campo dispondrá de 65536 posibles valores:

- Código marca: indica el código de la marca a la que pertenece el producto.
- Código producto: indica el código de ese producto.
- Código único: indica un código único de producto, para poder diferenciar productos iguales.

Para que los motes realicen las funciones deseadas, primero hay que cargarles un código con los eventos y tareas que se desea que lleven acabo. Para poder cargar este código, los motes tienen que estar conectados al puerto USB del PC, desde el cual mediante unos comandos introducidos en la consola Cygwin

se instalará este código en el mote. La plataforma de desarrollo Cygwin se instala al realizar la instalación del sistema operativo TinyOS (véase Anexo A).

A continuación se van a mostrar las principales características del código en nesC cargado en los motes. Primeramente se puede distinguir entre el código cargado en los motes adheridos a cada producto y el mote receptor conectado al PC.

El código cargado en los motes adheridos a cada producto es el mismo para todos, lo único que cambia es la información que se envía en el campo *data* y que es específica de cada producto. En la Fig. 4.3 se muestra el evento encargado de crear el paquete con los datos del producto y de llamar al evento RadioSend que es el encargado de enviar el mensaje.

```
event result_t Timer.fired()
{
      prueba->addr=0;
      prueba->type=1;
      prueba->group=TOS_AM_GROUP;
      prueba->length=6;
      prueba->data[0]=cod_empresa1;
      prueba->data[1]=cod_empresa2;
      prueba->data[2]=cod_producto1;
      prueba->data[3]=cod_producto2;
      prueba->data[4]=cod_unico1;
      prueba->data[5]=cod_unico2;

      call RadioSend.send(0,6,prueba);

      return SUCCESS;
}

event result_t RadioSend.sendDone(TOS_MsgPtr sent, result_t success)
{
      return SUCCESS;
}
```

Fig. 4.3 Funciones para el envío de mensajes

También hay que añadir que para que la base de datos lleve un control actualizado sobre los productos en stock, éstos han de enviar el mensaje que se ha mostrado anteriormente cada cierto tiempo. Se ha creído conveniente que este tiempo sea de 5 minutos, ya que de esta manera se lleva un control actualizado que se ajusta bastante a la realidad y también se consigue aumentar el periodo de duración de las baterías, gracias ya de por si al bajo consumo de estos dispositivos y al espacio de tiempo entre envío de mensajes. Esta variable es definida cuando se llama a la función Timer.start y las unidades en las que hay que definirlo son los milisegundos, véase Fig. 4.4.

```
command result_t StdControl.start()
{
      call RadioControl.start();
      call Timer.start(TIMER_REPEAT, 18000000);

      return SUCCESS;
}
```

Fig. 4.4 Funciones para el envío de mensajes

En la Fig. 4.5 se muestra un diagrama de flujo con el funcionamiento de los motes adheridos a los productos.

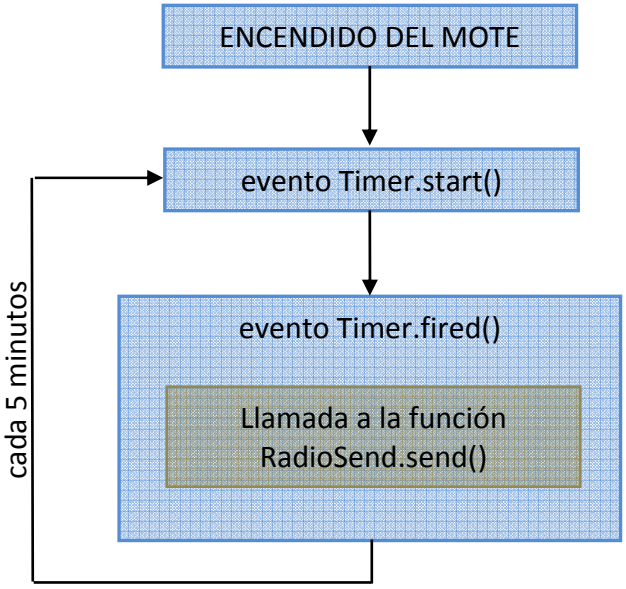

Fig. 4.5 Diagrama de flujo del mote de los productos

Por otro lado, con respecto al código cargado en el mote receptor, hay que destacar dos eventos:

a) el encargado de recibir los mensajes y almacenarlos en una cola, cuyo tamaño es definida por el usuario (véase Fig. 4.6).
b) el encargado de enviar estos paquetes almacenados en la cola del mote por el puerto USB para que los pueda procesar la aplicaciónPC (véase Fig. 4.7).

```
event TOS_MsgPtr RadioReceive.receive(TOS_MsgPtr Msg)
{
    TOS_MsgPtr pBuf = Msg;

    dbg(DBG_USR1, "TOSBase received radio packet.\n");

    atomic
    {
```

```
        if (!uartFull)
        {
            pBuf = uartQueue[uartIn];
            uartQueue[uartIn] = Msg;

            if( ++uartIn >= UART_QUEUE_LEN )
            {
                 uartIn = 0;
            }
            if (uartIn == uartOut)
            {
                 uartFull = TRUE;
            }
            if (!uartBusy)
            {
                 if (post UARTSendTask())
                 {
                      uartBusy = TRUE;
                 }
            }
        }
        else
        {
            dropBlink();
        }
    }
    return pBuf;
}
```

Fig. 4.6 Función para la recepción de mensajes

```
event result_t UARTSend.sendDone(TOS_MsgPtr msg, result_t success)
{
    if (!success)
    {
         failBlink();
    }
    else
    {
        atomic
        {
             if (msg == uartQueue[uartOut])
             {
                  if( ++uartOut >= UART_QUEUE_LEN ) uartOut = 0;
                  if (uartFull)
                  {
                       uartFull = FALSE;
                  }
             }
        }
    }

    post UARTSendTask();

    return SUCCESS;
}
```

Fig. 4.7 Función para el envío de mensajes a través del puerto USB

Finalmente se va a mostrar un esquema gráfico del funcionamiento del mote receptor (véase Fig. 4.8).

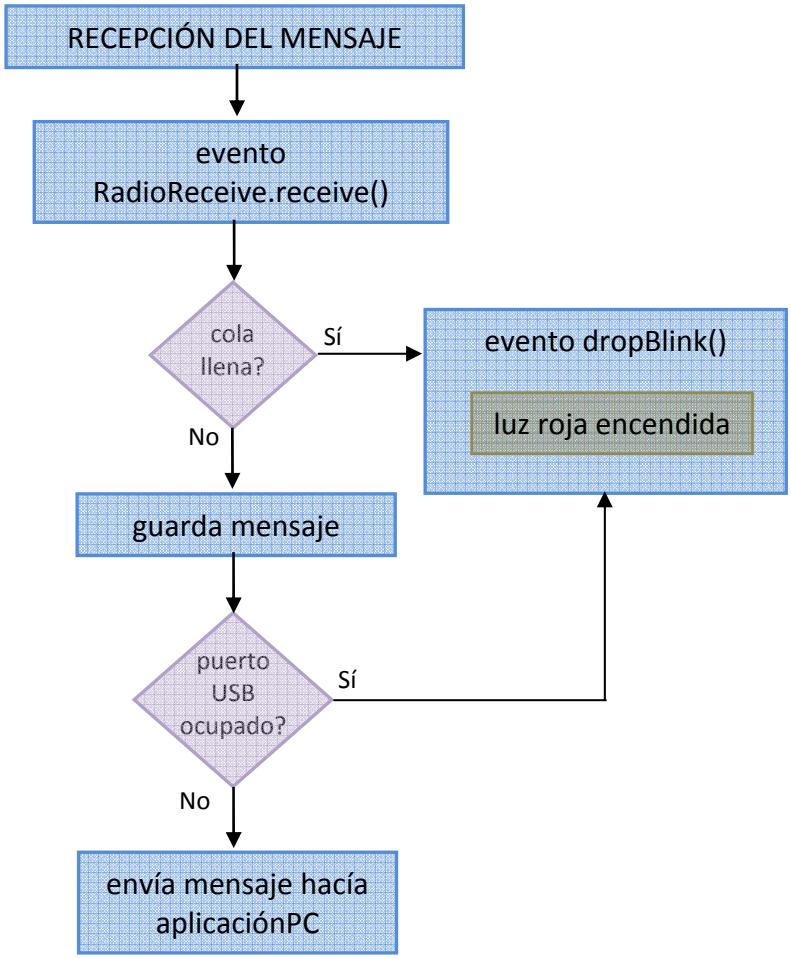

Fig. 4.8 Diagrama de flujo del mote receptor

4.3. Aplicación PC

La aplicaciónPC se ha desarrollado mediante la plataforma de desarrollo "OpenSource" EasyEclipse en su distribución EasyEclipse Server Java, la cual ha sido instalada sobre el sistema operativo Windows XP.

Esta aplicación consta de dos partes. Una de las partes es la encargada de procesar los mensajes recibidos por el mote receptor y almacenar esta información en la base de datos, concretamente en la tabla "info_producto", en la cual se guardará el código de marca, el código de producto y el código único que vienen dentro del mensaje como también la hora en la que se ha procesado ese mensaje, la cual la proporciona el sistema del PC.

Para que esta parte de la aplicación se ejecute, desde la pantalla principal de la aplicación (véase Fig. 4.9) habrá que clickar sobre el botón “Iniciar”, si en cualquier momento se desea detener el proceso habrá que clickar sobre el botón “Parar”.

Fig. 4.9 Pantalla principal de la aplicaciónPC

A continuación se muestra un diagrama de flujos con el funcionamiento de esta parte de la aplicación (véase Fig. 4.10), en el cual se puede observar como desde el momento que se clicka el botón “Iniciar”, la aplicación empieza a procesar todos los mensajes que el mote receptor le envía por el puerto USB. Se ha creado una estructura que contiene los mismos campos que el mensaje que se envía por medio radio, muy útil a la hora de almacenar la información y compararla con la información ya almacenada en la base de datos.

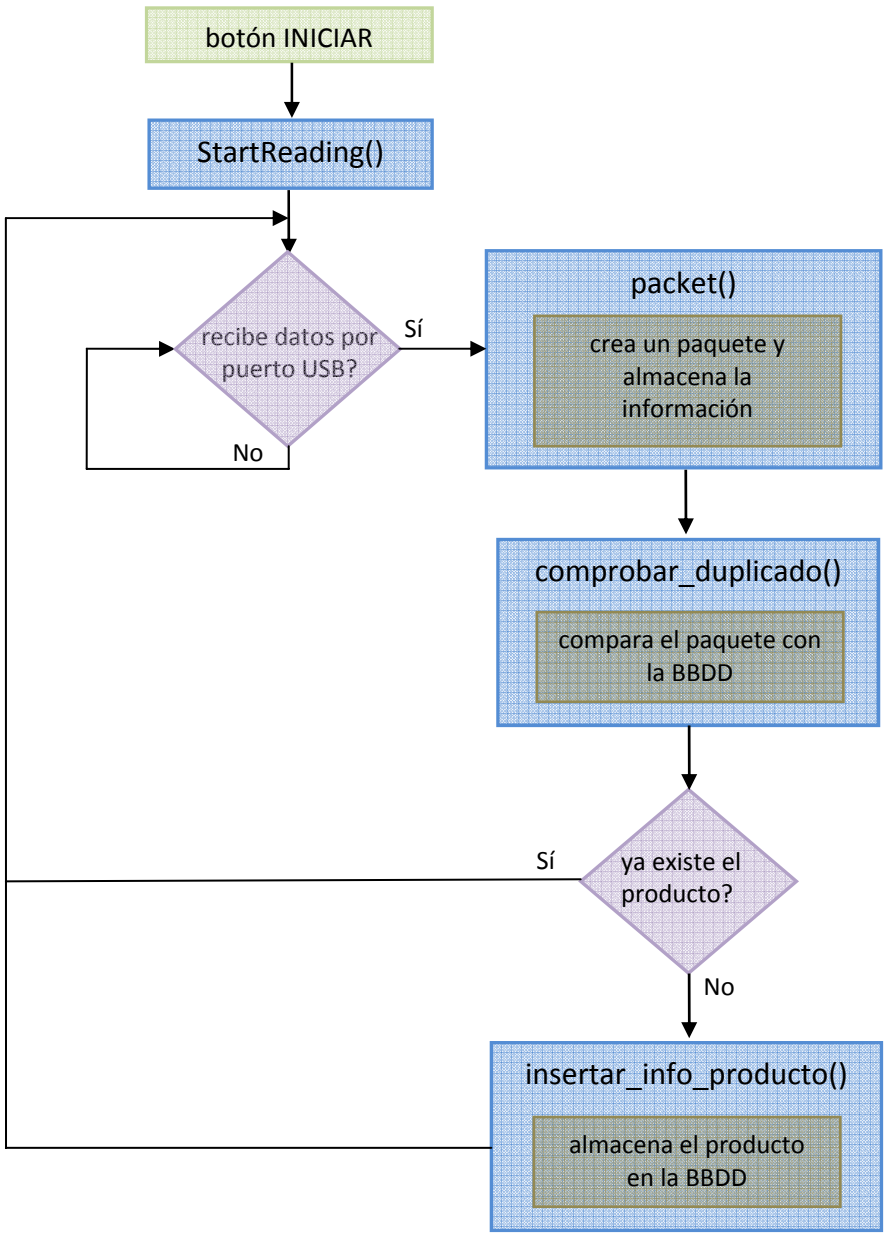

Fig. 4.10 Diagrama de flujo del procesado y almacenamiento de mensajes

Para acceder a la otra parte de la aplicaciónPC habrá que clickar sobre el botón “Gestor”, y aparecerá la pantalla que se muestra en la Fig. 4.11.

Fig. 4.11 Pantalla del Gestor de datos

Desde aquí según los privilegios que se tenga, dependiendo de si el usuario es administrador, usuario registrado ó usuario sin registrar, se tendrá acceso a ciertas operaciones, las cuales se muestran a continuación.

- Administrador
  - Se podrá dar de alta, modificar o eliminar a administradores.
  - Se podrá dar de alta, modificar o eliminar a usuarios.
  - Se podrá dar de alta, modificar o eliminar a marcas.
  - Se podrá dar de alta, modificar o eliminar a productos.

- Usuario registrado
  - Se podrá añadir productos al stock del almacén que por motivos ajenos a la aplicación no son detectados por ésta (ej. rotura del mote).
  - Se podrá eliminar productos del stock del almacén.

- Usuario sin registrar
  - Se podrá actualizar la tabla de la base de datos que contiene los productos en stock del almacén sin haber de esperar la actualización automática que realiza la aplicación cada 15 minutos.
  - Se podrá visualizar una tabla donde aparecen todos los productos en stock del almacén que existen actualmente. Antes de mostrar la tabla con los productos se realiza una actualización de la tabla de la base de datos que contiene los productos en stock.

A continuación se muestra mediante diagramas de flujo a que funcionalidades se tiene acceso dependiendo de los privilegios, en la Fig. 4.12 se muestran las funcionalidades de los administradores, en la Fig. 4.13 de los usuarios registrados y por último en la Fig. 4.14 de los usuarios sin registrar.

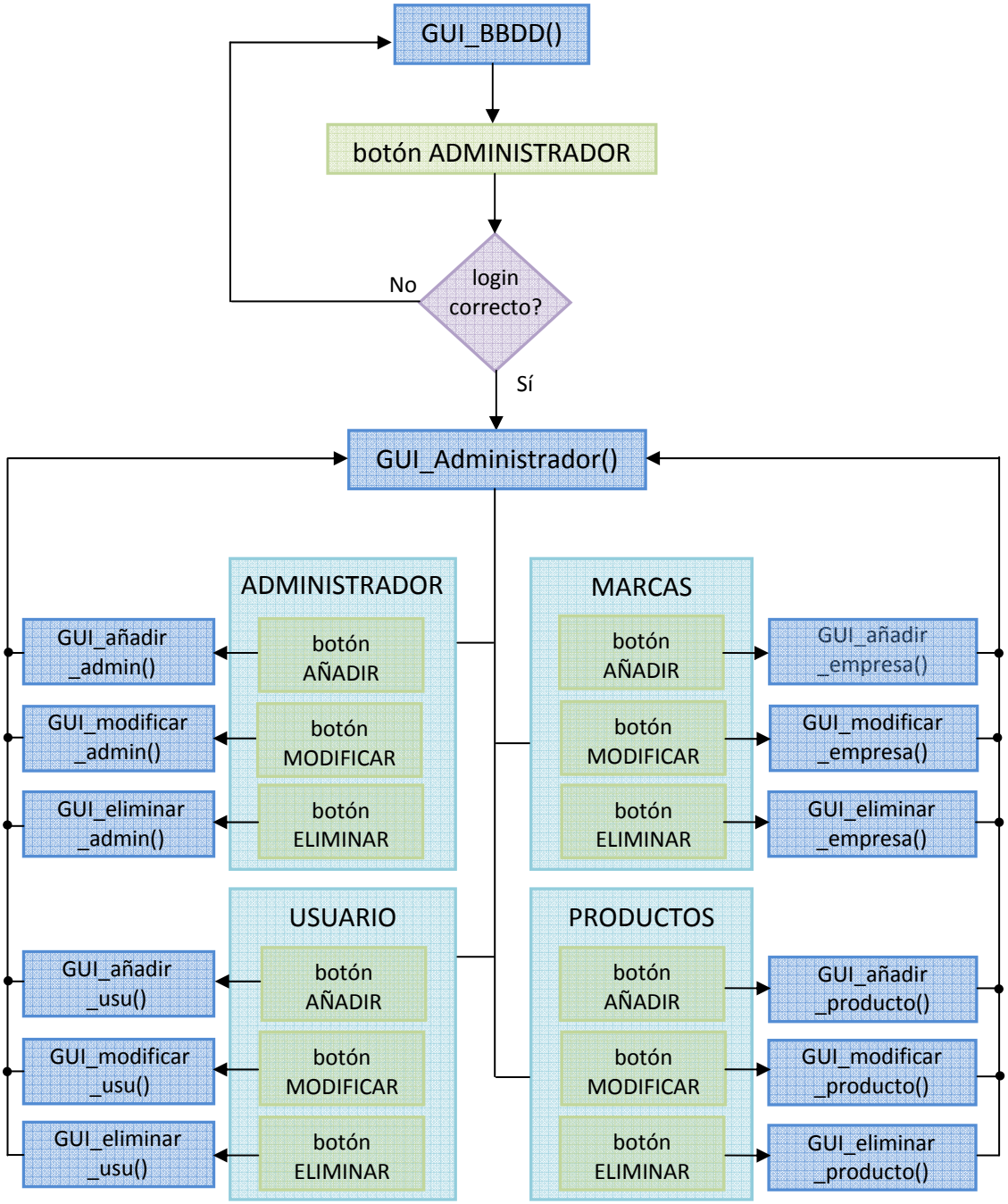

Fig. 4.12 Diagrama de flujo de los administradores

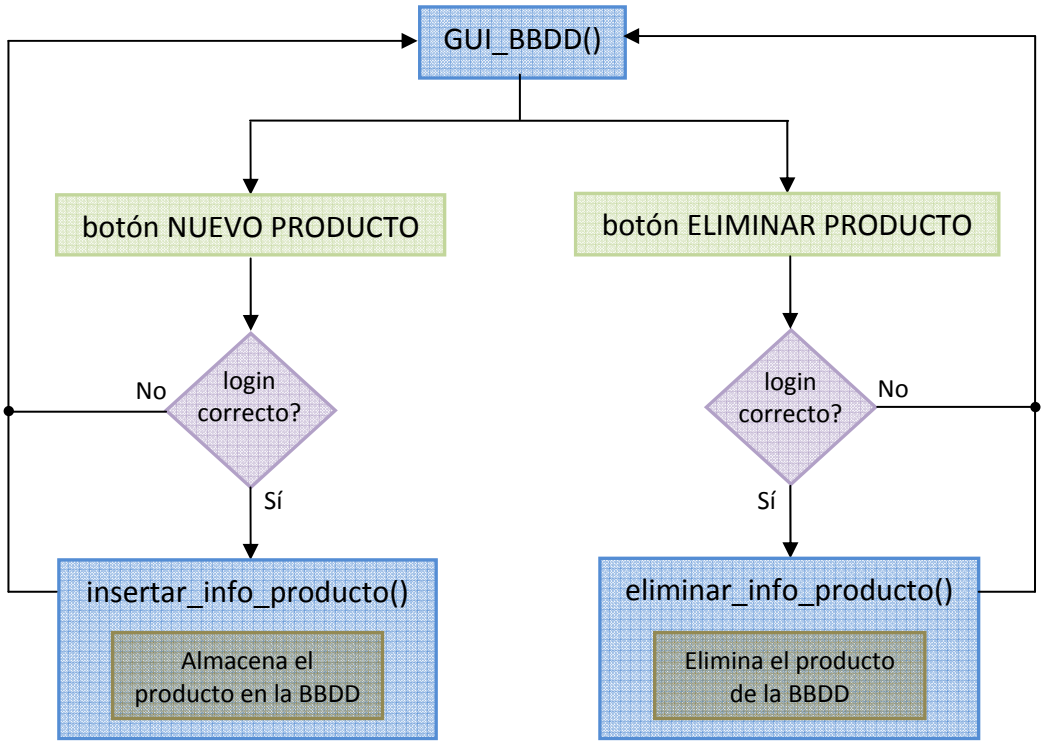

Fig. 4.13 Diagrama de flujo de los usuarios registrados

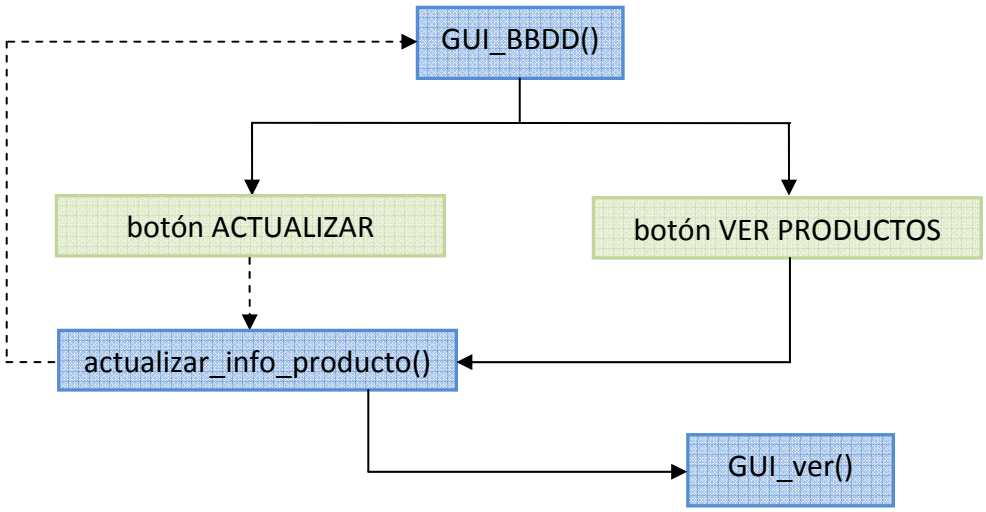

Fig. 4.14 Diagrama de flujo de los usuarios sin registrar

4.4. Aplicación móvil

La aplicación móvil se ha desarrollado mediante la plataforma de desarrollo “OpenSource” EasyEclipse en su distribución EasyEclipse Mobile Java, la cual ha sido instalada sobre el sistema operativo Windows XP.

En este apartado se va a realizar una descripción de las funcionalidades de las que dispone la aplicación móvil y posteriormente una pequeña descripción de la comunicación entre el dispositivo móvil y el *servlet*.

La aplicación móvil, para este proyecto dispone de dos tipos de búsqueda de productos en stock en el almacén, por marca y por producto. Aunque también se ha dejado implementado otro tipo de búsqueda por fecha que durante este punto se detallaran sus particularidades. La Fig. 4.15 muestra la pantalla inicial de la aplicación móvil. Añadir que desde cualquier otra pantalla se puede acceder a esta directamente o salir de la aplicación.

Fig. 4.15 Pantalla inicial de la aplicación móvil

Si se elige la búsqueda por marca, la aplicación pedirá al usuario el código de la marca del producto de la cual quiere realizar la búsqueda y si se disponen de productos en stock de esta marca, serán mostrados al usuario por la pantalla del dispositivo móvil, donde se indicarán los códigos de producto, el número de productos de ese tipo que se disponen y el nombre de los productos (véase Fig. 4.16). Añadir que si por el contrario en ese momento no se dispone en el almacén de ningún producto de la marca solicitada, se mostrará un mensaje en la pantalla del dispositivo móvil indicándolo.

Fig. 4.16 Búsqueda por marca

Por el contrario, si se elige la búsqueda por producto, la aplicación pedirá al usuario tanto el código de la marca del producto como el código del producto del cual se quiere realizar la búsqueda, en este caso al usuario se le mostrará por pantalla un mensaje indicándole el número de productos que se encuentran disponibles actualmente en stock (véase Fig. 4.17). Igual que en el caso anterior, si no se dispone en el almacén de ningún producto con el código solicitado, se mostrará un mensaje en la pantalla del dispositivo móvil indicándolo.

Fig. 4.17 Búsqueda por productos

Como se ha comentado al principio de este punto, en la aplicación móvil también se ha implementado otro tipo de búsqueda, en este caso por fecha, por si en un futuro se desea dotar a las etiquetas adheridas a los productos con más información como puede ser la fecha de entrada del producto en el almacén o la fecha de caducidad de éste. Para más información sobre las funcionalidades de la aplicación móvil, consultar el Anexo E.

A continuación se muestra mediante la Fig. 4.18, las funcionalidades descritas anteriormente de la aplicación móvil.

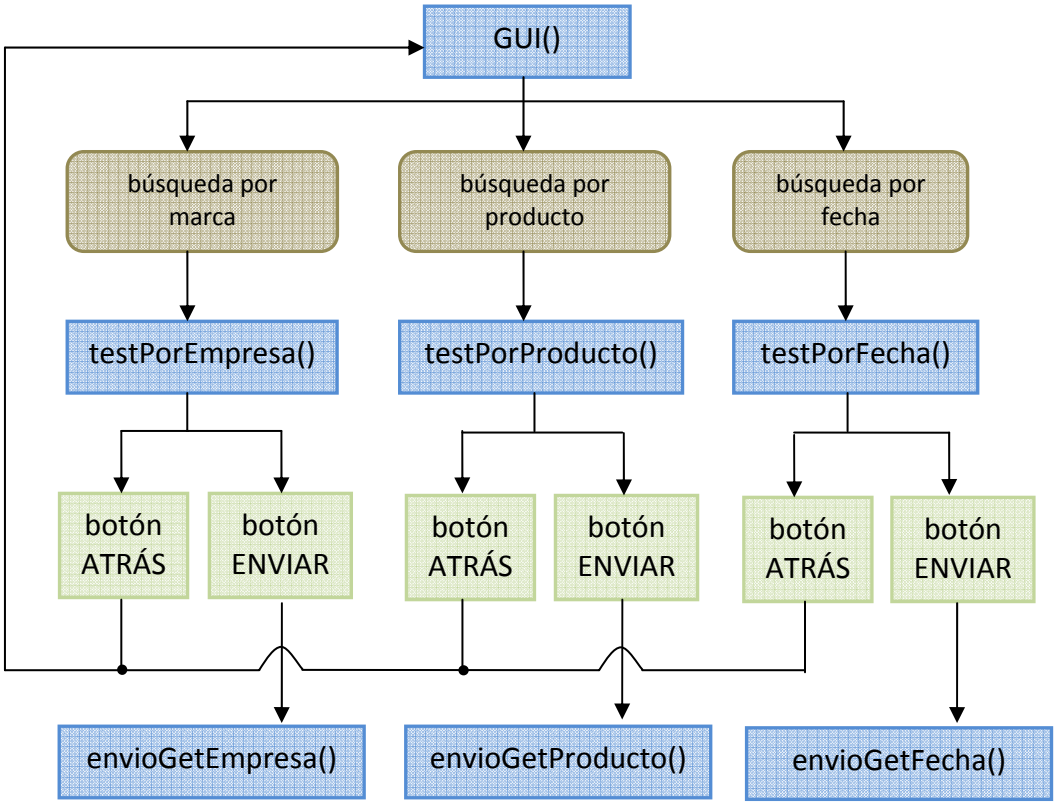

Fig. 4.18 Diagrama de flujo de la aplicación móvil

La Fig. 4.19 muestra un esquema de la comunicación entre la aplicación móvil y el *servlet* que se ejecuta dentro de un servidor, en nuestro caso Tomcat [17].

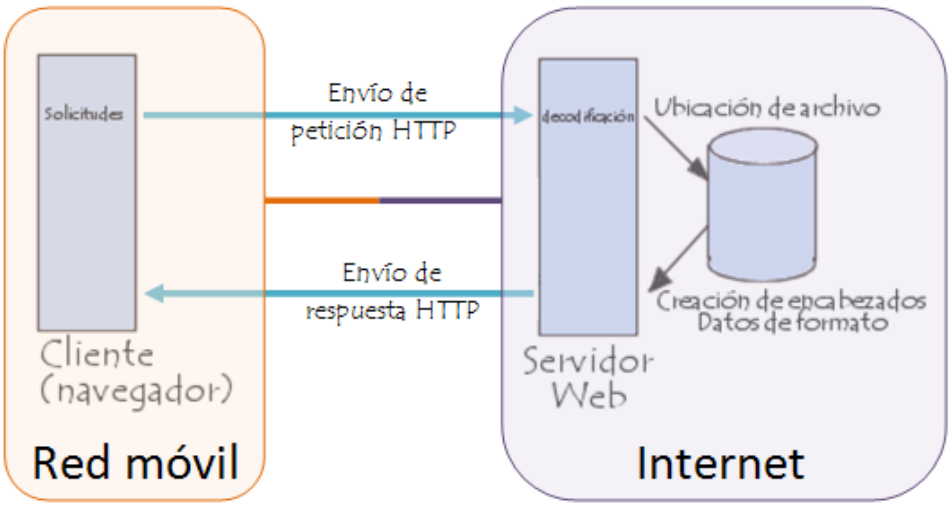

Fig. 4.19 Comunicación cliente-servidor web

El cliente, mediante la aplicación instalada en su móvil, introducirá los parámetros que se le pidan según la búsqueda que haya elegido y enviará una petición hacía el servidor web donde indicará el método utilizado, la URL, y la versión del protocolo, entre otros (véase Fig. 4.20). Una vez el servidor web haya recibido está petición y si la solicitud se ha llevado a cabo de manera correcta, enviará la respuesta hacía el dispositivo móvil, indicando la versión del protocolo utilizada, el código del estado y su significado (si la operación se ha efectuado correctamente será un “200 OK”) y la información que se le ha solicitado.

```
String url = "http://147.83.118.50:8080/infomobile/GetServlet" + "?" +
"empresa=" + input_producto_empre.getString() + "&" +"producto=" +
input_producto_produ.getString();
http = (HttpConnection)Connector.open(url);
http.setRequestProperty("Content-Language","es-ES");
http.setRequestProperty("User-Agent","Profile/MIDP-2.0
Configuration/CLDC-1.0");
http.setRequestProperty("Content-Type","application/octect-stream");
http.setRequestProperty("Connection","close");
http.setRequestMethod(HttpConnection.GET);
```

Fig. 4.20 Comandos utilizados en la petición

4.5. Service web

Como se ha explicado en el punto anterior, la aplicación móvil realiza las peticiones hacía un servidor web. En este punto se va a realizar una explicación más detallada de su funcionamiento. Primeramente decir que el nombre que se le ha dado a la aplicación que corre en nuestro servidor Tomcat es InfoMovile. Esta aplicación al recibir la petición de la aplicación móvil lo primero que hace es ver que parámetros se le han enviado para saber si el usuario ha realizado una búsqueda por marca o por producto.

Si el usuario de la aplicación móvil ha realizado una búsqueda por marca, éste ha tenido que introducir el código de la marca del producto que busca y este parámetro ha sido enviado dentro de la petición realizada. Una vez la aplicación InfoMovile sabe el tipo de búsqueda que se le ha pedido y los parámetros necesarios, realiza una consulta a la tabla de la base de datos donde se encuentran los productos en stock del almacén y guarda en un vector los productos pertenecientes a esa marca, el número de productos de cada tipo que hay y el nombre de esos productos. Finalmente esta información es enviada hacía la aplicación móvil y procesada por ésta.

Si por el contrario el usuario de la aplicación móvil ha realizado una búsqueda por producto, éste ha tenido que introducir el código del producto que busca y el código de la marca a la que pertenece. Igual que en el caso anterior la aplicación realiza una consulta a la base de datos y almacena en un vector de dos posiciones el número de productos que hay en stock del producto

solicitado y el nombre de ese producto. Finalmente esta información es enviada hacía la aplicación móvil y procesada por ésta.

4.6. Base de datos

4.6.1. Descripción BBDD

Como ya se ha explicado en el apartado 2.4 se ha elegido MySQL como el sistema de gestión de la base de datos. Éste ha sido instalado sobre el mismo PC donde se encuentra instalada la aplicaciónPC y el servidor, y cuyo sistema operativo es Windows XP.

A continuación se muestra un ejemplo de código en el cual se puede observar como se crean las tablas que son utilizadas para almacenar los datos proporcionados mediante la aplicaciónPC, en este ejemplo se muestra el código de la tabla que almacenará los posibles administradores y sus respectivas contraseñas (véase Fig. 4.21).

```
CREATE TABLE `administradores` (
  `ID` int(10) NOT NULL AUTO_INCREMENT,
  `ADMIN` varchar(10) NOT NULL,
  `ADMIN_PASS` varchar(10) NOT NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

Fig. 4.21 Código tabla administradores de la BBDD

En la Fig. 4.22 se muestran las diferentes tablas creadas para almacenar toda la información necesaria para el correcto funcionamiento de las aplicaciones diseñadas en este proyecto.

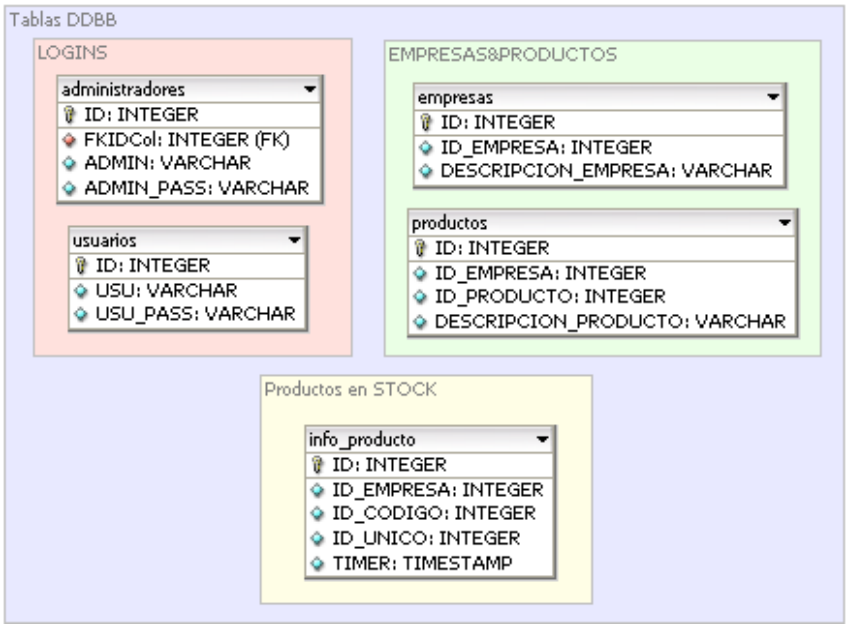

Fig. 4.22 Tablas de la BBDD

Las tablas “administradores” y “usuarios” son las encargadas de almacenar tanto a los administradores como a los usuarios registrados en la aplicación respectivamente, a los cuales se les permitirá tener acceso a ciertas funciones explicadas en el punto 4.3 como dar de alta a un nuevo usuario (en el caso que seas administrador) o añadir un producto al stock del almacén (en el caso que seas usuario registrado). Estas tablas están formadas por tres campos diferentes, el primero de ellos es un identificador, el segundo es el campo donde se almacena el *login* y por último el campo donde se almacena el *password* de cada *login*.

La tabla “empresas” se encarga de almacenar el código de la marca de los productos y el nombre de todas las marcas con las que la empresa FERNÁNDEZ S.A. tiene contratos.

En la tabla “productos” se almacenan todos los productos que la empresa FERNÁNDEZ S.A. puede tener en stock. Para poder añadir un nuevo producto a esta tabla, primero se ha tenido que dar de alta a la empresa suministradora de éste. Esta tabla esta formada por los campos identificador de empresa en la cual irá el código de la empresa que suministra el producto, el identificador de producto donde irá el código de producto y por último la descripción del producto donde irá el nombre del producto.

Hay que añadir que sólo los administradores registrados en la tabla “administradores” de la base de datos tienen privilegios para poder realizar modificaciones en estas tablas a la hora de añadir, modificar o eliminar. También decir que sólo se pueden realizar estos cambios desde la aplicaciónPC.

Por último la tabla “info_producto” almacena todos los productos que el almacén tiene en stock. En esta tabla se almacena el código de la marca, el código del producto y un código único de cada producto para poder realizar una diferenciación entre productos iguales. Estos tres datos son proporcionados por el mote TelosB adherido al producto. Un cuarto campo “Timer” es rellenado por la aplicaciónPC al procesar el mensaje enviado por los motes en el cual se introduce la hora del sistema del PC en el momento del procesamiento del mensaje.

Esta tabla es actualizada cada 15 minutos por la aplicaciónPC para que los productos almacenados en la tabla correspondan en la medida de lo posible a los productos que hayan en el almacén, además también puede ser actualizada manualmente y mostrada por pantalla por usuarios sin registrar que tengan acceso a la aplicaciónPC. Sólo esta aplicación es la que puede realizar cambios en las entradas de dicha tabla.

También hay que añadir que la aplicación “InfoMovile” puede acceder a la tabla “info_producto” pero sólo para realizar consultas pedidas por usuarios desde la aplicación móvil.

4.6.2. Interacción Java – MySQL

A continuación se va mostrar las líneas de código en java utilizadas para añadir, modificar o eliminar datos en la base de datos utilizada en este proyecto.

```
  stmt.executeUpdate("INSERT INTO info_producto (ID_EMPRESA, ID_CODIGO,
                          ID_UNICO, TIMER) VALUES
('"+co_empresa+"','"+co_producto+"','"+co_unico+"','"+sqlTimestamp+"')");
```

Fig. 4.23 Añadir un elemento a una tabla de la BBDD

En la línea de código que aparece en la Fig. 4.23 se puede observar como se utiliza el comando INSERT INTO seguido del nombre de la tabla donde se quiere insertar el elemento y finalmente el nombre de cada campo y el valor correspondiente.

```
          stmt.executeUpdate("UPDATE administradores SET
        ADMIN_PASS='"+password+"' WHERE ADMIN='"+login+"'");
```

Fig. 4.24 Modificar un elemento a una tabla de la BBDD

En este caso lo que se muestra en la Fig. 4.24 es cómo realizar una modificación de algún elemento que ya se encontraba en la base de datos. Para ello hay que utilizar el comando UPDATE, seguido del nombre de la tabla donde se quiere modificar el elemento y finalmente el nombre del campo con el

nuevo valor y el nombre de otro campo con su respectivo valor para que la aplicación tenga una referencia para saber dónde debe realizar la modificación.

```
stmt.executeUpdate("DELETE FROM administradores WHERE
                    ADMIN='"+login+"'");
```

Fig. 4.25 Eliminar un elemento a una tabla de la BBDD

Finalmente en la Fig. 4.25 se muestra la línea de código necesaria para eliminar una entrada de una tabla de la base de datos, para ello se ha de utilizar el comando DELETE, seguido del nombre de la tabla donde se quiere realizar la eliminación y el nombre de un campo con su respectivo valor para proporcionar a la aplicación una referencia de que entrada debe eliminar.

CAPÍTULO 5. PRUEBAS REALIZADAS

Este capítulo pretende validar las dos aplicaciones desarrolladas, así como explicar detalladamente las pruebas realizadas para comprobar su correcto funcionamiento.

Hay que destacar que todas las pruebas realizadas se han ejecutado sobre un ordenador con las siguientes características técnicas:

- Procesador: AMD Sempron 64 3000+ a 1.81GHz
- RAM: 512 MB DDR
- Sistema Operativo: Microsoft Windows XP Home
- USB 2.0 (High Speed)

5.1. Pruebas del servicio web

Para dar fe del buen rendimiento del servicio web, éste ha sido sometido a pruebas de stress mediante la herramienta de libre distribución JMeter. JMeter es una herramienta de carga para llevar acabo emulaciones sobre cualquier recurso de software. Mediante esta herramienta se van a realizar peticiones HTTP emulando a usuarios móviles realizando búsquedas de productos.

Su funcionamiento es muy sencillo. Primero hay que indicarle los siguientes tres parámetros (véase Fig. 5.1):

- Número de *threads*: Equivale al número de usuarios que se desean emular.

- Periodo de subida: Es el lapso de tiempo en segundos que se desea tener entre cada grupo de usuarios ("*Thread Group*").

- Contador del bucle: Utilizado para indicar el número de veces que se va a llevar acabo la emulación.

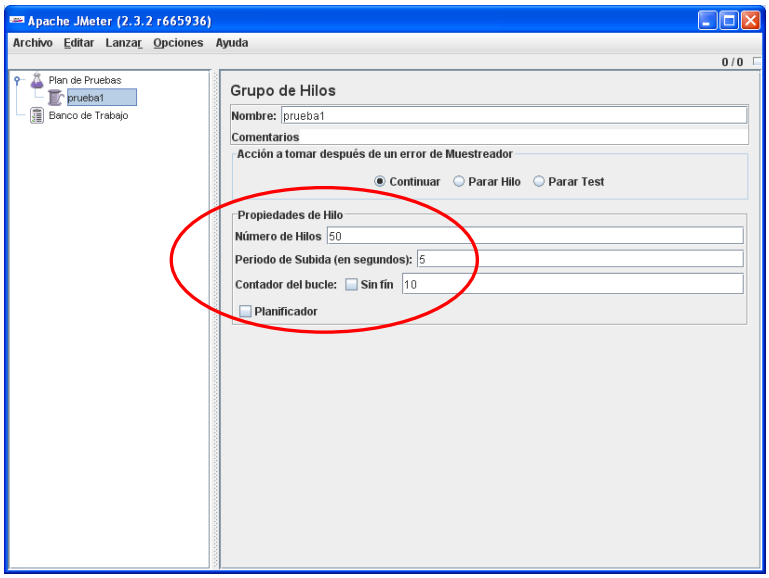

Fig. 5.1 Pantalla de configuración del *Thread Group*

Y una vez definidas las características del *Thread Group*, hay que indicarle la dirección del servidor al cual van a ir dirigidas las peticiones HTTP (véase Fig. 5.2):

- Nombre del servidor o IP: Empleado para definir la dirección IP o nombre del servidor donde será llevada a cabo la prueba de carga.
- Puerto: define el puerto TCP de operación del servidor.
- *Path*: Utilizado para definir la ruta de acceso para llevar acabo la prueba.

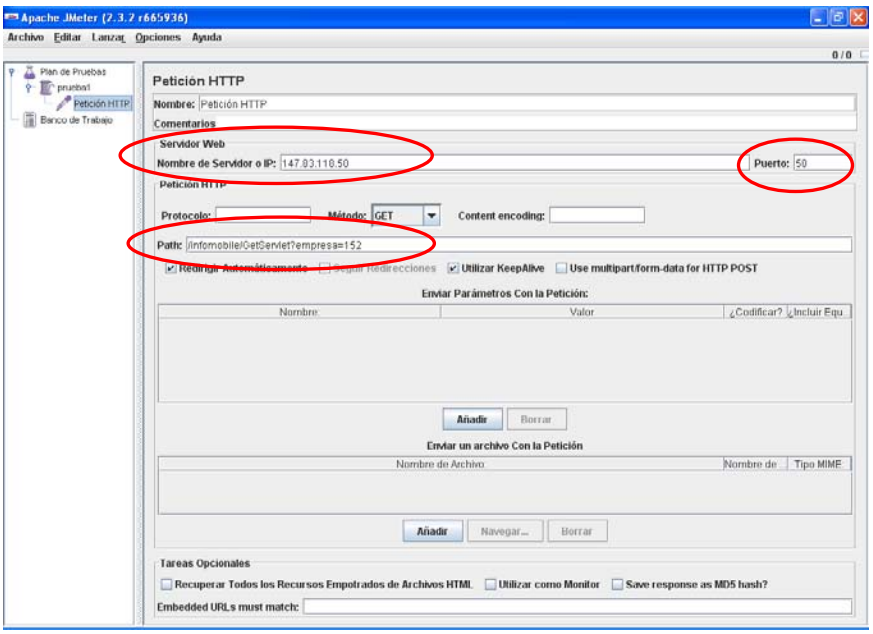

Fig. 5.2 Pantalla de configuración del *HTTP Request*

Para el proyecto se ha decidido realizar 3 pruebas de stress, todas ellas emulando peticiones HTTP y aumentando el número de usuarios. El número de usuarios se ha ido aumentando progresivamente en las 3 pruebas, donde la variable que diferencia las pruebas es el tiempo entre ciclos que los usuarios tienen para realizar su petición, en la Fig. 5.3 el tiempo de ciclo es de 1 segundo, en la Fig. 5.4 es de 5 segundos y en la Fig. 5.5 es de 10 segundos. Lo que se quiere saber con estas pruebas es el porcentaje de peticiones realizadas erróneas que nos podemos encontrar y así llegar a saber los límites que puede tener nuestro servidor.

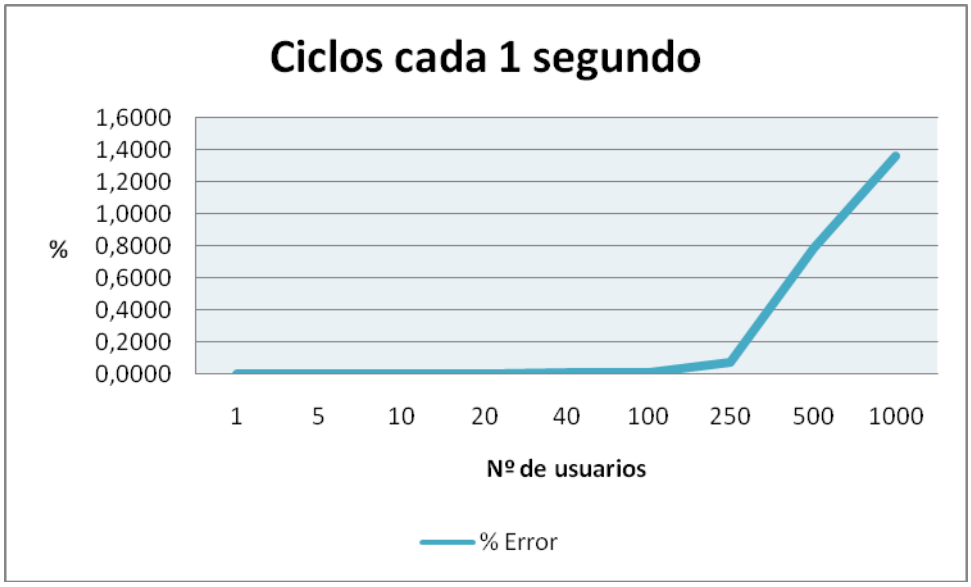

Fig. 5.3 %Error con 1 segundo entre ciclos

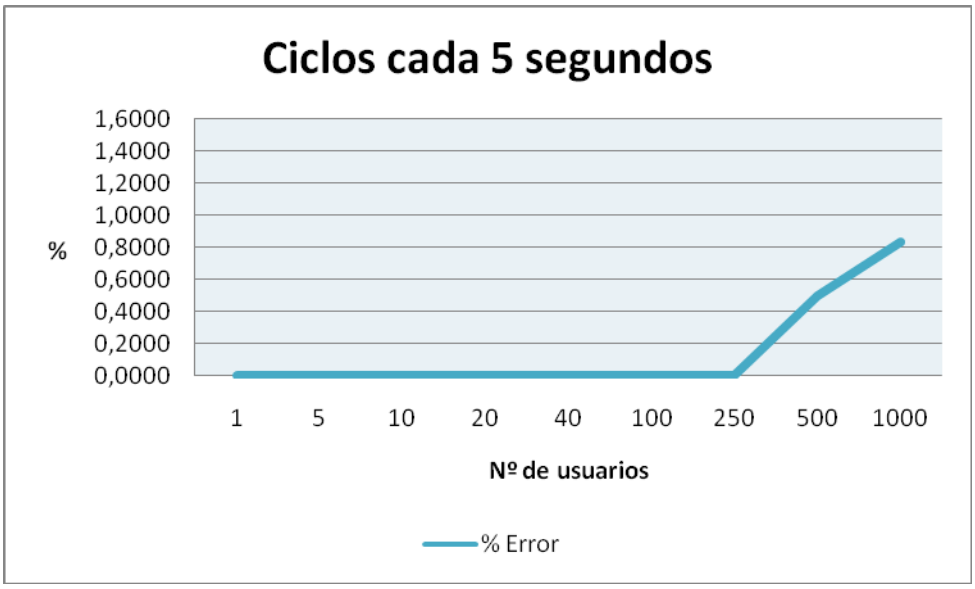

Fig. 5.4 %Error con 5 segundos entre ciclos

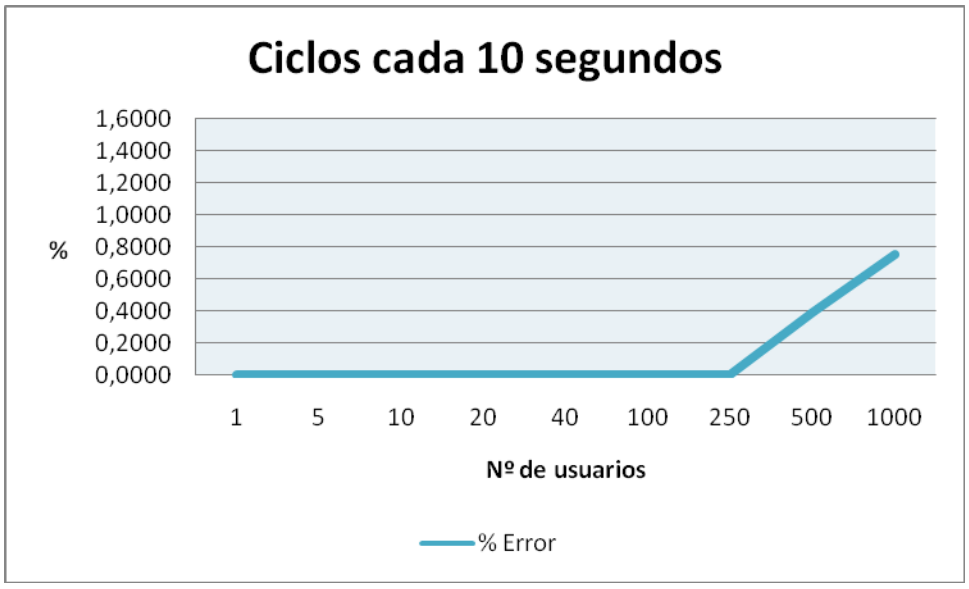

Fig. 5.5 %Error con 10 segundos entre ciclos

Como conclusiones extraídas de estas pruebas realizadas, se puede decir que la carga permitida por el servidor es correcta para nuestra aplicación, ya que es muy poco probable que la aplicación sea utilizada por 1000 usuarios simultáneamente, y aunque sucediera, el % de error obtenido de peticiones no procesadas es muy bajo.

5.2. Pruebas de validación

En este punto se va a mostrar el correcto funcionamiento entre la aplicaciónPC y la base de datos. Para ello se ha utilizado el grid que hay instalado en el laboratorio 325 de la Escola Politècnica Superior de Castelldefels. El grid consiste en una estructura de madera, colgada mediante cable de nylon a 1 metro del techo. El número total de dispositivos inalámbricos TelosB de que se disponen son 60, puesto que el grid consta de 10 filas por 6 columnas. En cada cruceta de la cuadrícula hay situado un mote, y entre ellos existe casi un metro de distancia (90 centímetros exactamente).

En la Fig. 5.6 se puede apreciar un esquema visual de la estructura del grid.

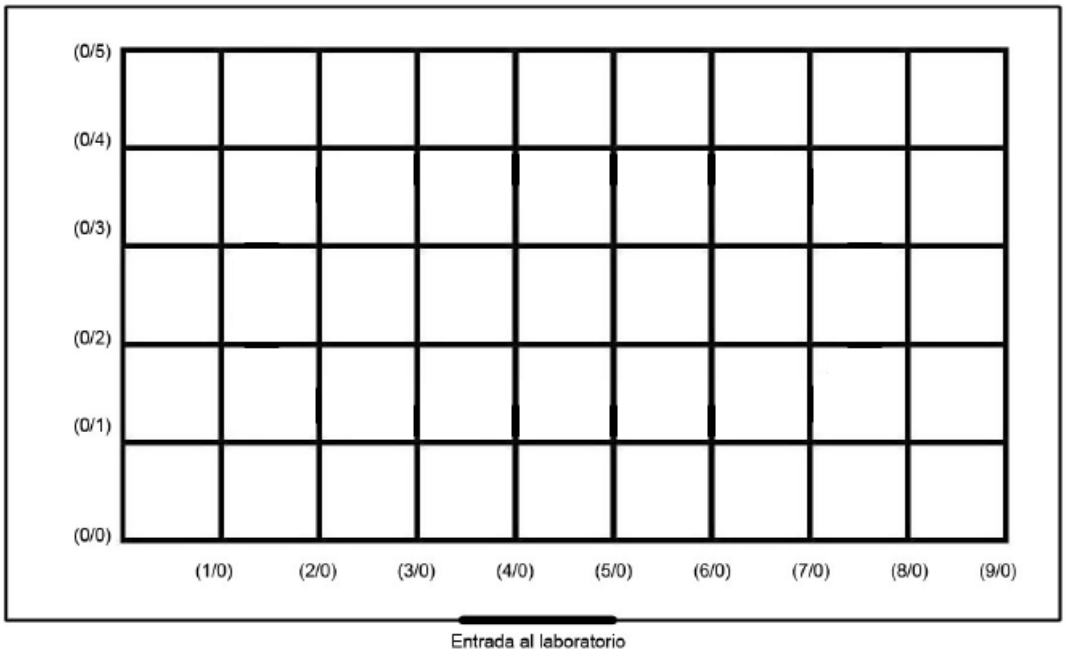

Fig. 5.6 Esquema del grid

En la Fig. 5.7 se muestra una fotografía real de la cuadrícula del laboratorio y de los motes situados en cada cruceta.

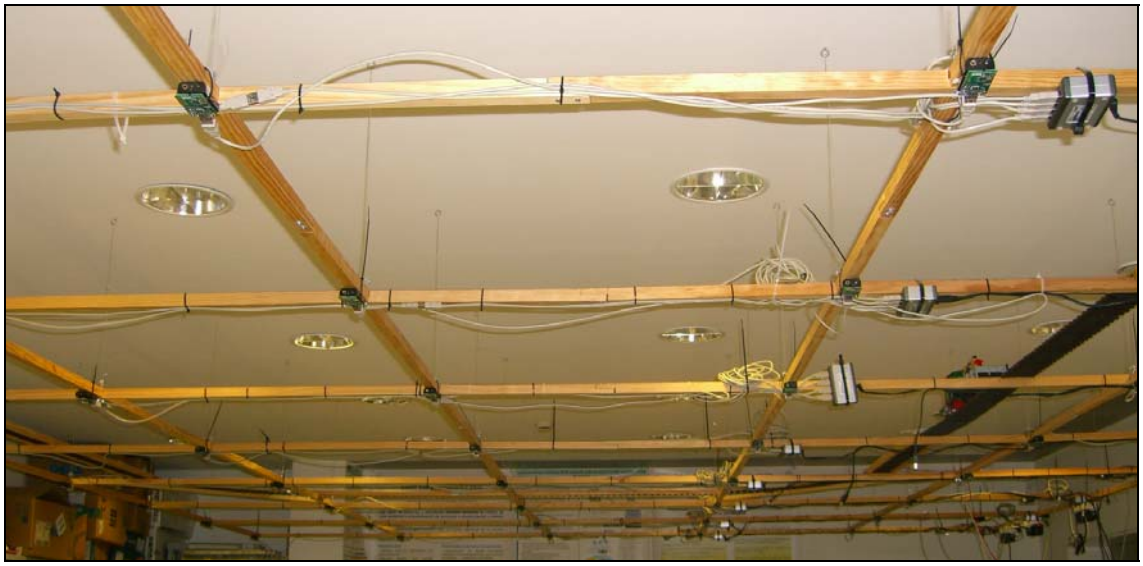

Fig. 5.7 Fotografía de la estructura del grid

Para realizar esta prueba se instaló en cada mote un código de producto, habiendo productos de las mismas marcas e incluso productos iguales donde la única diferencia era su código único. Añadir que los motes trabajan en el canal 26 ya que es en el que menos interferencias se producen con otras tecnologías inalámbricas que también transmiten a la frecuencia de 2.4 GHz. Otra variable muy importante es la que define la potencia a la que transmiten

los sensores, éstos transmitirán a la máxima potencia (0 dBm) ya que es con la única que cuando había una distancia considerable (a partir de 10 metros) la señal atravesaba paredes y se establecía comunicación entre el mote receptor y el mote del producto.

Posteriormente se inicializó la aplicaciónPC y ésta empezó a procesar todos los mensajes recibidos por el mote receptor. Hay que decir que en esta prueba se redujo el tiempo entre envío y envío de los mensajes de los 5 minutos definidos en el apartado 4.2 a 15 segundos, por lo tanto para comprobar su correcto funcionamiento sólo se tuvo que esperar este periodo de tiempo y ver mediante la aplicaciónPC, la tabla que muestra los productos en stock del almacén (véase Fig. 5.8), añadir que la aplicación recibió y procesó correctamente la información enviada por todos los motes. También para corroborar este correcto funcionamiento, se hizo un seguimiento a la tabla info_producto de la base de datos y se pudo ver como para cada producto a los 15 segundos de haber enviado su último mensaje, se actualizaba el campo TIMER de la tabla info_producto con la hora actual del sistema.

Gracias a esta prueba se ha comprobado también, que es posible llevar un control de los productos actualizado próximo al tiempo real, ya que con el intervalo de 15 segundos ha funcionado todo correctamente.

Código marca	Código producto	Descripción
5654	262	Satellite L300
5398	259	Aspire 5530
2571	2827	FULLHD 42"
2828	2827	SELPHY ES2
2826	2827	Cineos FLAT TV 47"
3083	2827	Stylus S20
5398	260	Aspire 5930
5398	258	TravelMate 5720
2827	2827	32LG5600
5654	517	Satellite A200
771	514	impresora K8600
771	514	impresora K8600
771	514	impresora K8600
5398	259	Aspire 5530
5398	259	Aspire 5530

Fig. 5.8 Tabla de productos en stock

Para comprobar también que la base de datos eliminaba de la tabla info_producto a los productos que durante más de 15 minutos no recibía ningún mensaje, se realizó una segunda prueba muy sencilla, en la cual consistía en apagar un mote y de esta manera conseguir que dejara de enviar mensajes. Una vez apagado el mote se esperó el tiempo oportuno. En esta prueba, igual que en la anterior, el tiempo entre envío de mensajes de los motes es de 15 segundos, por lo tanto se esperó 45 segundos (15 segundos x 3 envíos permitidos).

En la Fig. 5.9 se muestra la tabla con los productos en stock y en la Fig. 5.10 se muestra la misma tabla pero pasados 45 segundos, se puede apreciar como el producto correspondiente al mote apagado, en este caso la impresora K5400 ha sido eliminado.

Fig. 5.9 Tabla de productos en stock

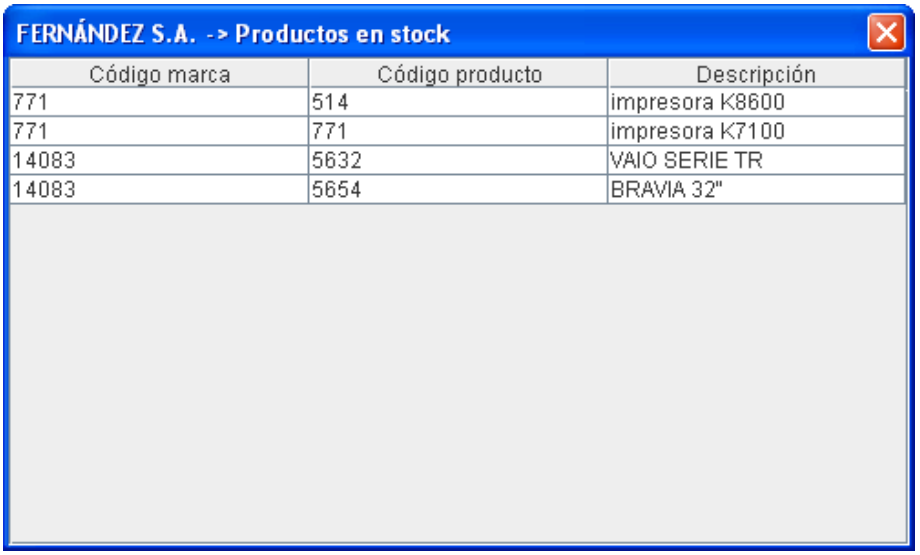

Fig. 5.10 Tabla de productos en stock pasados 45 segundos

CAPÍTULO 6. CONCLUSIONES Y LÍNEAS FUTURAS

A continuación se exponen las conclusiones extraídas del proyecto, así como diferentes opciones de expansión y mejoras del mismo.

6.1. Conclusiones

El interés por la tecnología RFID se ha incrementando con rapidez. Muchas empresas están buscando aumentar la eficiencia de sus operaciones y reducir costes a través de esta tecnología (p.ej. realización de inventarios). La oferta de este tipo de soluciones cada vez es mayor y se espera que en aproximadamente 10 ó 15 años la tecnología RFID sea utilizada de forma cotidiana, ya que el coste de uno de sus componentes, el *tag*, se ha reducido en los últimos años y para finales de 2008, se espera que el coste de las etiquetas llegue a US$0.05 cada una, lo cual permitirá el empleo masivo de esta tecnología y el surgimiento de muchas aplicaciones que aprovechen sus características.

El poder de RFID se encuentra principalmente en 3 cualidades: la capacidad de poder leer etiquetas a distancia y sin necesidad de línea visible con el receptor, la capacidad de lectura/escritura y el poder identificar a elementos como únicos. Estas características son claves y representan un gran diferenciador al comparar RFID con otras tecnologías de autoidentificación.

Los objetivos propuestos al inicio de este PFC se han cumplido satisfactoriamente.

Primero se ha llevado a cabo una aplicación capaz de almacenar en una base de datos la información enviada por los motes, y para comprobar su correcto funcionamiento se han utilizado 60 motes enviando mensajes a la vez, con un satisfactorio comportamiento por parte de la aplicación.

La segunda aplicación desarrollada permite a los usuarios que puedan realizar consultas a la base de datos mediante sus dispositivos móviles, y de esta manera saber en todo momento y desde cualquier punto geográfico los productos es stock que hay en el almacén, siempre y que se disponga de cobertura y acceso a Internet. Se han realizado unas pruebas de stress sobre el servidor, las cuales han dado unos resultados satisfactorios.

Una vez realizadas las pruebas de validación de cada parte del proyecto, se realizó una prueba conjunta en la que intervinieron los 60 motes de los que está compuesto el grid, enviando sus respectivos mensajes, la aplicaciónPC procesando esos mensajes y almacenándolos correctamente en la base de datos y finalmente realizando una petición HTTP desde el dispositivo móvil solicitando la existencia de un producto, obteniendo un resultado satisfactorio de la unión de las tres partes.

6.2. Líneas futuras de trabajo

A partir del trabajo realizado en este PFC, se abren nuevos caminos de desarrollo:

Para empezar se podría mejorar el código que ejecutan los motes para enviar los mensajes de manera que los motes puedan emplear un protocolo multisalto. De esta manera si un mote no se encuentra dentro del área de cobertura del mote receptor pueda llegarle la información referente al producto. El principal inconveniente de esta mejora es el aumento exponencial de mensajes con la consecuencia que todos los motes tengan más funcionalidades como procesar mensajes con su mayor consumo de batería.

Donde de verdad existen varias líneas futuras abiertas es en la aplicación móvil. La primera de ellas, como ya se ha dicho en el transcurso de la memoria, es la de dotar de más información a los productos como la fecha de caducidad si éste tuviera o la fecha de entrada del producto, y para ello ya se ha dejado desarrollada la búsqueda por fecha en la aplicación móvil pero habría que modificar la base de datos para dotar la tabla “info_producto” de más campos y modificar las funciones de búsqueda de la aplicación “InfoMobile”. Otra mejora sería la de crear un servicio de aviso mediante SMS, para ello se tendría de realizar una interfaz gráfica donde el usuario, por ejemplo pudiera elegir para que productos se quiere recibir la información o cuando se quiere recibir esta información (cuando el producto este agotado o queden “x” unidades). Ésta última mejora supondría un gasto adicional y mensual a la empresa ya que debe comprar los servicios a alguna empresa colaboradora de alguna operadora telefónica que le gestione por ejemplo un *gateway* SMPP[10].

[10] SMPP es un protocolo estándar pensado para el intercambio de mensajes SMS.

IMPLICACIONES MEDIOAMBIENTALES

A continuación se exponen los aspectos del Proyecto Final de Carrera que pueden suponer algún tipo de impacto medioambiental.

Para empezar, los sensores con los que se ha trabajado consumen baterías eléctricas, en nuestro caso, pilas del tipo AA. Si éstas no son depositadas en el contenedor correspondiente para su reciclado o correcto almacenamiento pueden causar estragos en los ecosistemas donde se depositen e incluso pueden llegar a afectar a los seres humanos por medio de la contaminación del agua y del aire.

Las pilas ofrecen una fuente de energía cómoda y portátil que forma parte integral de numerosos productos electrónicos modernos. Sin embargo, tan sólo son capaces de almacenar una pequeña parte de la energía que se emplea en su fabricación. Suponen una carga para el medio ambiente tanto en su fabricación como en su eliminación. Algunos de los componentes de las pilas, como el cadmio, mercurio o zinc, pueden ser gravemente perjudiciales para el medio ambiente y ocasionar enfermedades a los seres humanos (dolores gastrointestinales) si no reciben el tratamiento adecuado.

Conscientes de esto, los fabricantes de pilas han conseguido reducir notablemente los componentes peligrosos como el mercurio. Además, han comenzado a desarrollar tecnologías para recuperar los componentes de las pilas cuando éstas se agotan e incluso se han inventado las denominadas pilas verdes o biopilas, que son pilas que destacan por la ausencia de los metales pesados en ellas y por tanto por la poca contaminación que causa su consumo en el medioambiente.

En la misma línea, cabe destacar que ZigBee, IEEE 802.15.4, está diseñado para maximizar el tiempo de vida de las baterías. El mayor consumo que realizan estos dispositivos es a la hora de transmitir información, lo que conlleva una vida útil de batería muy elevada (en teoría, del orden de años).

Por otro lado, también hay que añadir que cada vez hay más demanda de nuevas tecnologías, esto ocasiona que cada vez se desarrollen mejores y más potentes aplicaciones que necesitan de medios físicos (tanto fijos como móviles) más potentes, esto provoca que el ciclo de vida de estos medios sea bastante corto y necesiten renovarse de forma continua favoreciendo el aumento de los residuos.

BIBLIOGRAFÍA

Referencias

[1] Concepto de RFID,
http://www.efalcom.com.ar/rfid/index.php?option=com_content&task=blogsection&id=4&Itemid=11

[2] Definición de RFID,
http://es.wikipedia.org/wiki/RFID

[3] Historia de RFID,
http://www.it.uc3m.es/jmb/RFID/rfid.pdf

[4] Definición del código de barras,
http://es.wikipedia.org/wiki/C%C3%B3digo_de_barras

[5] Ventajas de RFID vs código de barras,
http://www.ihg.net/java/X?cgi=lateral.rfid.VentajaVsBarras.pattern

[6] Datasheet motes TelosB,
http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf

[7] ZigBee/IEEE 802.15.4 Summary,
http://www.eecs.berkeley.edu/~csinem/academic/publications/zigbee.pdf

[8] IEEE 802.15 WPAN Task Group 4,
http://www.ieee802.org/15/pub/TG4.html

[9] ZigBee Alliance,
http://www.zigbee.org

[10] Plataforma Java2 y arquitectura J2EE,
http://www.jebussines.com:8080/peruvirtual/arquitectura.jsf

[11] Ciclos de vida de un MIDlet,
http://www.ciimurcia.es/informas/ene05/articulos/Moviles_Java_algo_mas_que_juegos.html

[12] Base de datos MySQL,
http://www.xtec.net/~acastan/textos/Administracion%20de%20MySQL.html#SERVIDOR

[13] Tecnologías de almacenamiento,
http://www.tufuncion.com/myisam-vs-innodb

[14] API JDBC,
http://camoralesm.googlepages.com/jdbc

[15] Tomcat,
http://es.wikipedia.org/wiki/Apache_Tomcat#Enlaces_externos

[16] Estructura de directorios Tomcat,
http://www.programacion.com/tutorial/tomcatintro/

[17] Comunicación cliente-servidor,
http://es.kioskea.net/internet/http.php3

Bibliografía

El soporte técnico utilizado para la programación de los motes en el lenguaje nesC han sido los siguientes portales:

- http://telegraph.cs.berkeley.edu/tinydb/nesdoc/mica/tos.lib.TinyDB.NetworkC.nc.html
- http://www.cs.wmich.edu/~gupta/teaching/cs603/wsnSp04/lec07b%20Active%20Messages%20IvanA%20020504.pdf

El soporte técnico utilizado para la programación de las aplicaciones en lenguaje Java han sido los siguientes:

Conocimientos básicos

- http://java.sun.com/j2se/1.5.0/docs/api/
- http://www.programacion.com/java/tutorial/java_basico/
- usuarios.lycos.es/manualesjava/manuales/javaCero/Manual.Curso.de.java.desde.cero.pdf
- http://edisoncor.wordpress.com/2007/10/05/crear-un-jbutton-redondeado-y-personalizado/
- http://www.iescamp.es/tutoriales/java/tema8/tema8p9_1.html
- http://ji.ehu.es/LMAlonso/SW/java/Bib/tutorjava/html/ui/swingcomponents/dialog.html
- http://www.chuidiang.com/chuwiki/index.php?title=JFrame_y_JDialog

Desarrollo de la aplicación móvil

- http://www.lcc.uma.es/~galvez/J2ME.html
- http://developers.sun.com/mobility/reference/codesamples/
- http://eclipseme.org/docs/index.html

Interactuación con la BBDD

- http://www.webestilo.com/mysql/
- http://www.chuidiang.com/java/mysql/mysql-java-basico.php

Servlet

- http://www.programacion.net/tutorial/servlets_basico/
- http://www.java2s.com/Code/Java/J2ME/InvokeServletMidlet1.htm

Plataforma TelosB

- Tutorial del Sistema Operativo TinyOS, http://www.tinyos.net/tinyos-1.x/doc/tutorial/
- Mensajes radio, http://www.chipcon.com/index.cfm?kat_id=1&action=faq&faq_id=340
- CC2420 (chip radio), http://www.chipcon.com/files/CC2420_Data_Sheet_1_0.pdf

Manual JMeter

- http://jakarta.apache.org/jmeter/usermanual/index.html
- http://www.osmosislatina.com/jmeter/basico.htm

Lista de Acrónimos

API: Application Programming Interface. Interfaz de Programación de Aplicaciones.

BBDD: Base de Datos.

ERO: European Radiocommunications Office.

CEPT: Conférence européenne des administrations des postes et des télécommunications. Conferencia Europea de Administraciones de Correos y Telecomunicaciones.

EPC: Electronic Product Code. Código Electrónico de Producto.

ETSI: European Telecommunications Standards Institute. Instituto Europeo de Normas de Telecomunicaciones

HF: High Frequency. Alta Frecuencia.

HTML: Hyper Text MarkUp Language. Lenguaje de Marcas de Hiper Texto.

HTTP: Hyper Text Transfer Protocol. Protocolo de Transferencia de Hiper Texto.

IEEE: Institute of Electrical and Electronics Engineers.

IP: Internet Protocol. Protocolo de Internet.

ISO: International Organization for Standardization. Organización Internacional para la Estandarización.

J2EE: Java 2 Enterprise Edition.

J2ME: Java 2 Micro Edition.

J2SE: Java 2 Standart Edition.

JAR: Java ARchive.

JDBC: Java DataBase Connectivity. Conectividad de Bases de Datos Java.

JDK: Java Development Kit. Kit de Desarrollo de Java.

JRE: Java Runtime Environment. Entorno en Tiempo de Ejecución de Java.

JVM: Java Virtual Machine. Máquina Virtual de Java.

LF: Low Frequency. Baja Frecuencia.

PC: Personal Computer. Ordenador Personal.

SQL: Structured Query Language. Lenguaje Estructurado de Consulta.

SMPP: Short Message Peer to Peer protocol. Protocolo de Mensajes Cortos Peer to Peer.

UHF: Ultra High Frequency. Frecuencia Ultra Alta.

URL: Uniform Resource Locator. Localizador de Recursos Uniforme.

TCP: Transmission Control Protocol. Protocolo de Control de Transmisión.

XML: Extensible Markup Language. Lenguaje Extensible de Marcas.

Escola Politècnica Superior de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANEXOS

TÍTULO DEL PFC: Sistema de gestión de productos con emulación de RFID mediante sensores IEEE 802.15.4

TITULACIÓN: Ingeniería de Telecomunicaciones (segundo ciclo)

AUTOR: Eloy Fernández López

DIRECTOR: Carles Gómez Montenegro

FECHA: 21 de octubre de 2008

ANEXO A. TinyOS

A.1. Instalación TinyOS para Windows

Requisitos para la instalación:

- El CD-ROM de CrossBow que contiene las herramientas de soporte de TinyOS.
- Ordenador con SO Microsoft Windows (XP, 2000, NT), con 1 GB de espacio libre en la unidad de disco de destino de la instalación.
- Tener los programas: WinZip, Adobe Acrobat y Programmers Notepad (disponible gratuitamente en http://www.pnotepad.org/).

La instalación de TinyOS para Windows es sencilla. En primer lugar se tiene que instalar el archivo ejecutable *tinyos-1.1.0-1is.exe,* el cual nos instalará la plataforma básica de desarrollo Cygwin, las librerías nesC de TinyOS y el compilador nesC.

El entorno de instalación de este ejecutable es muy intuitivo, guiado en todos sus pasos. Este archivo se puede encontrar en:

- El CD de CrossBow, dentro de la carpeta *TinyOS Install.*
- En el servidor de descargas de la plataforma Windows de www.tinyos.net que se encuentra en la siguiente página de internet http://www.tinyos.net/dist-1.1.0/tinyos/windows/

Después de la instalación de la plataforma TinyOS, la consola Cygwin y compilador de nesC, concentrados en el archivo *tinyos-1.1.0-1is.exe,* procederemos a la instalación del paquete rpm que nos actualizará a la última versión del compilador nesC. El nombre del paquete es el *nesc-1.1.2a-1.cygwin.i386.rpm* y se puede encontrar en el mismo servidor de descargas anterior (http://www.tinyos.net/dist-1.1.0/tinyos/windows/).

La instalación de este paquete se tiene que efectuar desde la consola Cygwin, dentro del directorio donde se encuentre el paquete rpm. La secuencia a escribir es la siguiente:

rpm -- force --ignoreos -Uvh nesc-1.1.2a-1.cygwin.i386.rpm

Finalmente se instalará la actualización de TinyOS que se desee, como requisito necesita tener instalado el anterior paquete. La actualización más reciente es la 1.1.15 de diciembre de 2005, y para descargarla sólo se tiene que ir al servidor de descargas de Windows ya indicado por duplicado anteriormente. El link a presionar es el tinyos-1.1.15Dec2005cvs-1.cygwin.noarch.rpm.

La instalación de este segundo paquete rpm se instalará de semejante forma al anterior paquete:

rpm -- force --ignoreos -Uvh tinyos-1.1.15Dec2005cvs-1.cygwin.noarch.rpm

La aplicación para los motes TelosB realizadas en este trabajo han sido realizadas con las versiones de TinyOS:

- tinyos-1.1.7July2004cvs-2.cygwin.noarch.rpm
- tinyos-1.1.10Jan2005cvs-1.cygwin.noarch.rpm

La carpeta xbow del CD de CrossBow dentro de la carpeta *TinyOS Updates se* copiará en la carpeta de nuestro disco C:\tinyos\cygwin\opt\tinyos-1.x\contrib. De esta forma se podrá comenzar a trabajar desde el directorio C:\tinyos\cygwin\opt\tinyos-1.x\contrib\xbow\apps que se ha copiado, programando nuestras aplicaciones en nesC.

Con el comando **rpm –qa** en la consola Cygwin se puede comprobar la correcta instalación de TinyOS y los paquetes de actualización.

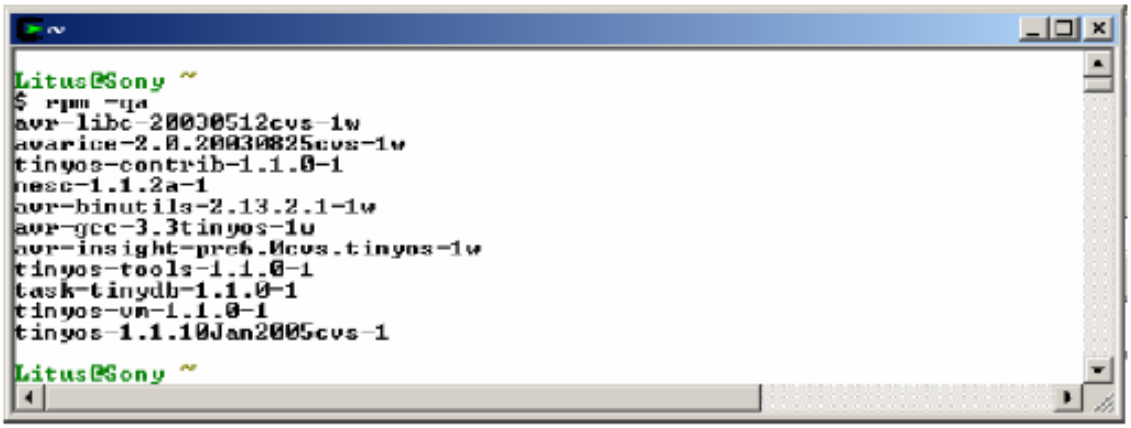

Fig. A.1 Ejemplo del comando de comprobación “rpm -qa”

A.2. Directorio TinyOS

Para trabajar con el directorio de TinyOS se utilizará la consola Cygwin que con los comandos de linux nos permite explorar el directorio.

El directorio de carpetas de TinyOS está compuesto de numerosas carpetas y archivos. Este apartado pretende mostrar el contenido de las carpetas más importantes de TinyOS y su estructura.

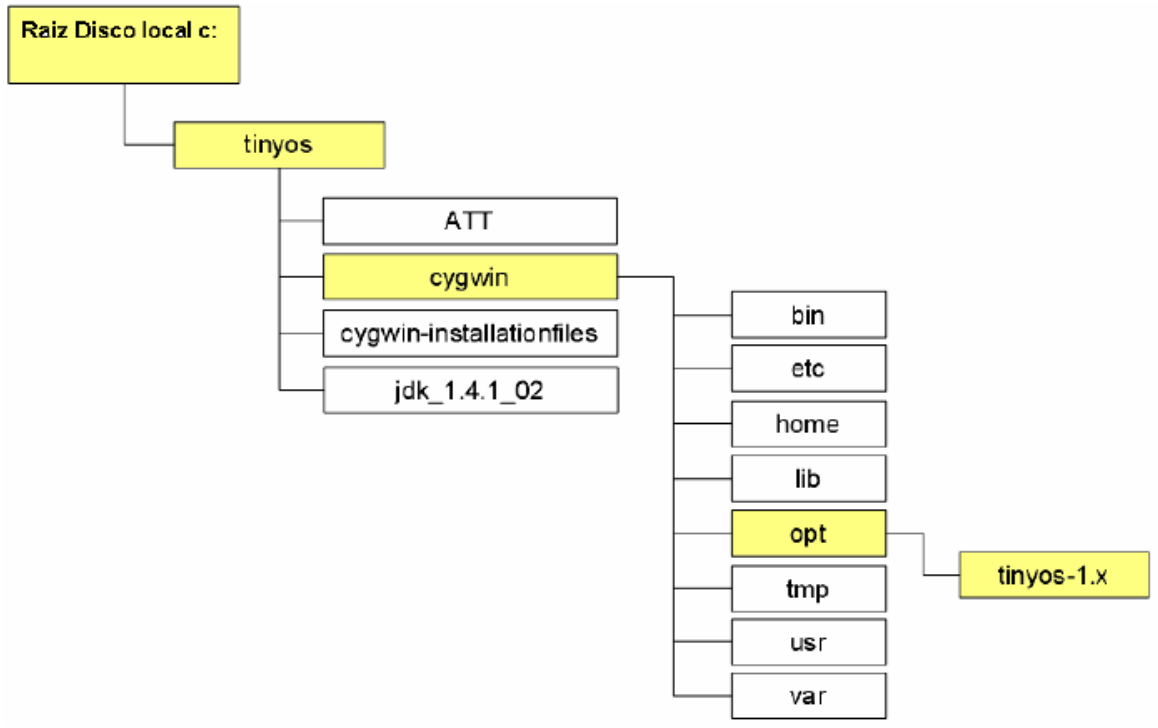

Fig. A.2 Estructura principal desde la raíz del directorio TinyOS

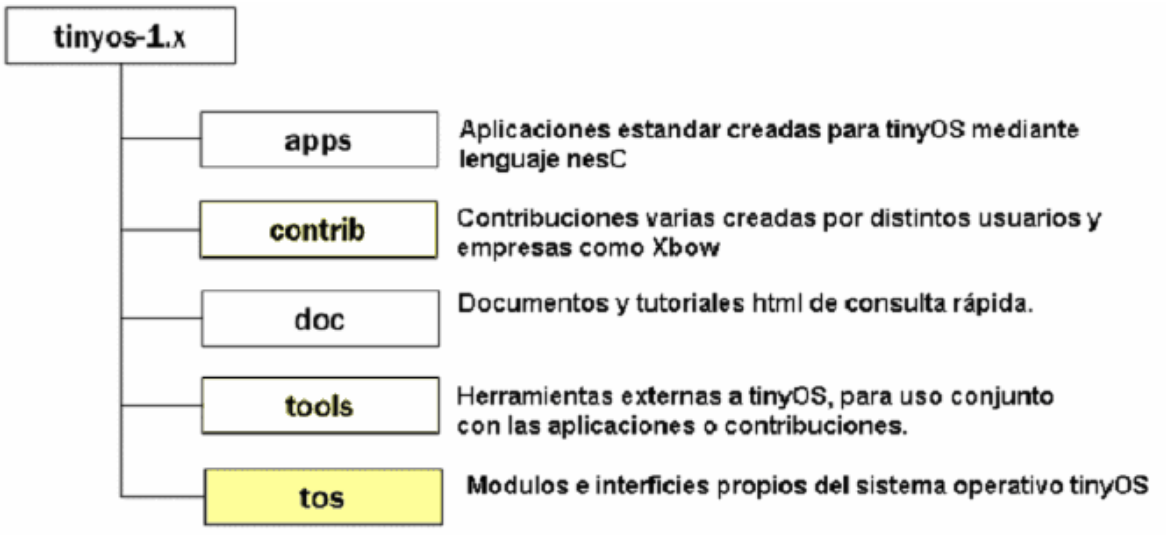

Fig. A.3 Estructura y contenido del directorio tinyos-1.x

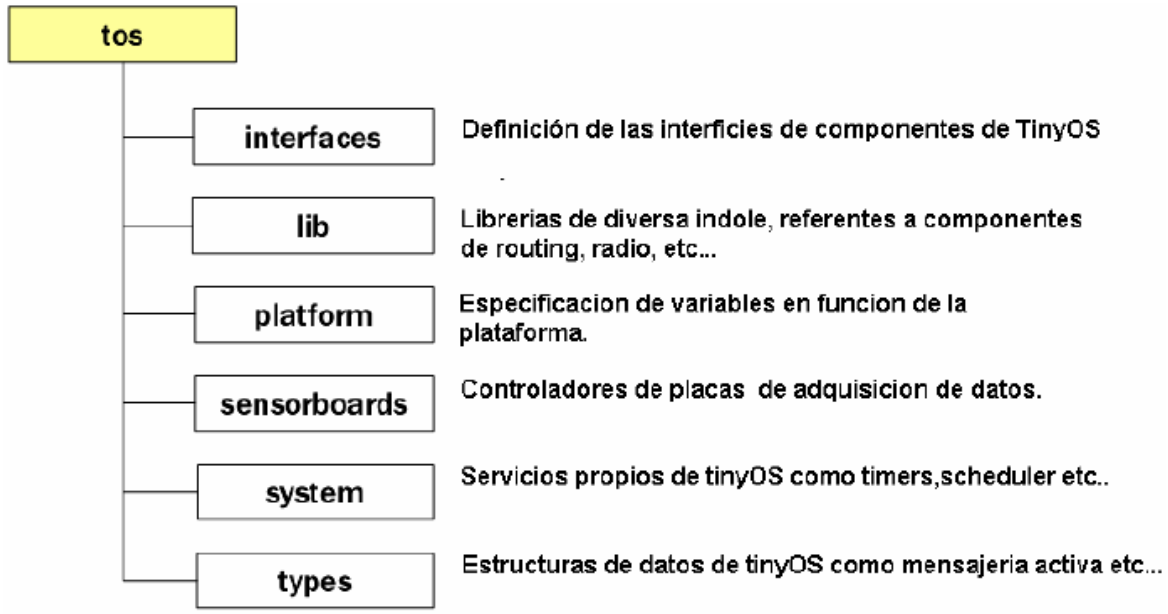

Fig. A.4 Estructura y contenido del directorio tos dentro de la carpeta tinyos-1.x

A.3. Compilación de aplicaciones en TinyOS

La sintaxis que se utiliza en la consola Cygwin para compilar las aplicaciones es la siguiente:

make <plataforma> install <dirección dispositivo> <programador>,<puerto>
o
make <plataforma> reinstall <programador>,<puerto>

donde:

- plataforma es el nombre de la plataforma, en este caso “telosb”.
- dirección dispositivo es un número que es proporcionado mediante UART protocol. Se utiliza el comando “motelist” para saber este número.
- programador es el nombre del microcontrolador, en este caso msp-430-bsl “bsl”, cuyas siglas quieren decir Boot Strap Loader.
- puerto es el “COM-1”, donde COM es la dirección del dispositivo.

La diferencia entre install o reinstall está detalla a continuación.

- Install.<n>: Compila la aplicación para la plataforma seleccionada, puede configurar el Identificador de nodo (<n>) del mote y lo programa.

- Reinstall.<n>: Puede también configurar el Identificador de nodo y instala el programa precompilado, no recompila. Esta opción es sustancialmente más rápida.

ANEXO B. Especificaciones técnicas TelosB

Specifications	TPR2420CA	Remarks
Module		
Processor Performance	16-bit RISC	
Program Flash Memory	48K bytes	
Measurement Serial Flash	1024K bytes	
RAM	10K bytes	
Configuration EEPROM	16K bytes	
Serial Communications	UART	0-3V transmission levels
Analog to Digital Converter	12 bit ADC	8 channels, 0-3V input
Digital to Analog Converter	12 bit DAC	2 ports
Other Interfaces	Digital I/O,I2C,SPI	
Current Draw	1.8 mA	Active mode
	5.1 µA	Sleep mode
RF Transceiver		
Frequency band[1]	2400 MHz to 2483.5 MHz	ISM band
Transmit (TX) data rate	250 kbps	
RF power	-24 dBm to 0 dBm	
Receive Sensitivity	-90 dBm (min), -94 dBm (typ)	
Adjacent channel rejection	47 dB	+ 5 MHz channel spacing
	38 dB	- 5 MHz channel spacing
Outdoor Range	75 m to 100 m	Inverted-F antenna
Indoor Range	20 m to 30 m	Inverted-F antenna
Current Draw	23 mA	Receive mode
	21 µA	Idle mode
	1 µA	Sleep mode
Sensors		
Visible Light Sensor Range	320 nm to 730 nm	Hamamatsu S1087
Visible to IR Sensor Range	320 nm to 1100nm	Hamamatsu S1087-01
Humidity Sensor Range	0-100% RH	Sensirion SHT11
Resolution	0.03% RH	
Accuracy	± 3.5% RH	Absolute RH
Temperature Sensor Range	-40°C to 123.8°C	Sensirion SHT11
Resolution	0.01°C	
Accuracy	± 0.5°C	@25°C
Electromechanical		
Battery	2X AA batteries	Attached pack
User Interface	USB	v1.1 or higher
Size (in)	2.55 x 1.24 x 0.24	Excluding battery pack
(mm)	65 x 31 x 6	Excluding battery pack
Weight (oz)	0.8	Excluding batteries
(grams)	23	Excluding batteries

Notes
[1]Programmable in 1 MHZ steps, 5 MHz steps for compliance with IEEE 802.15.4/D18-2003.
Specifications subject to change without notice

ANEXO C. Configuración parámetros del chip CC2420

A continuación se muestran y explican los parámetros más importantes de los dos archivos de configuración básicos que definen las variables del chip CC2420.

C.1. Archivo de configuración AM.h

En este apartado se pueden ver las sentencias más importantes definidas en el archivo de configuración AM.h con sus valores por defecto.

```
#define DEF_TOS_AM_GROUP 0x7d  //define el valor del campo grupo. Por
                                 defecto 125

#define TOSH_AM_LENGTH 1      //define el valor del campo grupo. Por
                                defecto toma valor 1

typedef struct TOS_Msg        //Estructura que define los campos del
                                mensaje TOS
{
Los siguientes campos son los que se transmiten y/o se reciben por
medio radio

  uint8_t length; //Este campo indica cuantos bytes enviamos en el
                    campo datos.

  uint8_t fcfhi; //Los campos fcfhi y fcflo son un registro de 16 bits
                 (parte hi y lo) donde cada grupo de bits indican una
                 característica de la transmisión (utilización ack o
                 el tipo de trama utilizada, etc).

  uint8_t fcflo;

  uint8_t dsn;    //Secuencia de paquetes enviados, en cada nuevo
                  paquete se incrementa en 1, este campo es rellenado
                  por el sistema TinyOS.

  uint16_t destpan;     //Indica la dirección de la red de área local a
                          la que va destinado el paquete.

  uint16_t addr;    //Indica la dirección del mote al que va enviado el
                    paquete.

  uint8_t type;   //Indica el tipo de paquete que enviamos.

  uint8_t group;  //Indica a que grupo pertenece el mote receptor

  int8_t data[TOSH_DATA_LENGTH];    //   Campo   donde   metemos   la
                                    información   útil   que   queremos
                                    transmitir

Los siguientes campos no se transmiten y/o reciben por medio radio.
```

```
  uint8_t strength; //Este campo nos indica la potencia de recepción.

  uint8_t lqi;    //Indica la calidad del enlace por el cual ha
                  recibido un determinado paquete.

  bool crc; //Este campo nos indica si el crc es correcto.

  uint8_t ack;    //Indica si recibimos reconocimiento de una
                  determinada trama.

  uint16_t time;  //Referencia interna del procesador.
} TOS_Msg;
```

Estos campos no son modificables una vez nos encontramos en medio de una comunicación por lo que toman el valor definido en este archivo de configuración.

C.2. Archivo de configuración CC2420Const.h

Otro archivo muy importante es el CC2420Const.h, donde por ejemplo se decide en que canal se quiere transmitir. En nuestro caso se ha decidido trabajar en el canal 26 ya que es en el que menos interferencias se producen con otras tecnologías inalámbricas que también transmiten a la frecuencia de 2,4 GHz. Otra variable muy importante es la que define la potencia a la que se quiere que transmitan los sensores, después de realizar diferentes pruebas, se ha decidido que se necesitaba que transmitieran a la máxima potencia (0 dBm) ya que era con la única que cuando había una distancia considerable (a partir de 10 metros) la señal atravesaba paredes y se establecía comunicación entre los motes.

A continuación se muestran los campos definidos en este archivo que también se considera importante para nuestro trabajo.

```
// Tiempos para el CC2420 en microsegundos
enum {
  CC2420_TIME_BIT = 4,
  CC2420_TIME_BYTE = CC2420_TIME_BIT << 3,
  CC2420_TIME_SYMBOL = 16
};

//En este campo definimos la potencia de recepción de la antena
CC2420. En este caso seleccionamos 0x1f que es 0 dBm, la potencia
maxima.
#define CC2420_DEF_RFPOWER          0x1f

//Definimos el canal por el que vamos a transmitir o recibir.
#define CC2420_DEF_CHANNEL        26

//Definimos la banda de frecuencia que vamos a utilizar, en este caso
seleccionamos la banda libre (2.405 GHz).
#define CC2420_DEF_PRESET           2405
```

```
//Definimos el campo de control de trama donde podemos controlar
cuestiones como el tipo de trama o si queremos o no el envío de ack.
#define CC2420_DEF_FCF_LO           0x08
#define CC2420_DEF_FCF_HI           0x01  // sin ACK
#define CC2420_DEF_FCF_HI_ACK       0x21  // con ACK
#define CC2420_DEF_FCF_TYPE_BEACON 0x00
#define CC2420_DEF_FCF_TYPE_DATA   0x01
#define CC2420_DEF_FCF_TYPE_ACK    0x02
```

ANEXO D. GUÍA DE USUARIO DE LA APLICACIÓN PC

En este anexo se va a explicar como utilizar la aplicaciónPC junto con todas las funciones que puede realizar esta aplicación.

D.1. Pantalla inicial de la aplicaciónPC

En la imagen de la Fig. D.1 se muestra la pantalla inicial de la aplicación, esta pantalla se puede dividir en cuatro zonas, las cuales se explican a continuación.

Fig. D.1 Pantalla inicial de la aplicaciónPC

- Zona 1: en esta zona se encuentran las pestañas de “dispositivos” y “ayuda”. La pestaña de dispositivos permite elegir el puerto USB deseado si se disponen de más de un mote receptor conectado al PC.

- Zona 2: en esta zona se encuentra el botón “Gestor”, el cual sirve para acceder a la parte de la aplicación desde la cual se gestionan todos los usuarios, marcas y productos.

- Zona 3: en esta zona se encuentra el botón “Iniciar”, el cual sirve para encender la parte de la aplicación encargada de procesar los paquetes recibidos por el mote receptor y llevar un control de los productos en stock en la base de datos.

- Zona 4: en esta zona está situado el botón “Salir”, el cual sirve para cerrar por completo la aplicación.

D.2. Procesado de los mensajes recibidos por el mote receptor

Una vez visto las cuatro zonas de la pantalla principal nos centraremos en explicar el funcionamiento de las dos partes más importantes de esta aplicación. Por un lado para que la base de datos tenga un control a tiempo real de los productos en stock que se encuentran en el almacén, se debe inicializar la parte de la aplicación encargada de procesar todos los paquetes recibidos por el mote receptor, para ello hay que clickar sobre el botón “Iniciar” que se encuentra en la pantalla inicial de la aplicación (véase Fig. D.2).

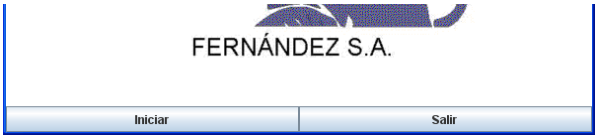

Fig. D.2 Botón “Iniciar”

Una vez inicializado este proceso, la aplicación se encarga de comparar la información que recibe con la información que se encuentra en la base de datos, si recibe información de un producto nuevo, el cual no se encuentra en la tabla correspondiente a los productos en stock de la base de datos, la aplicación ya se encarga de almacenar este nuevo producto. Si por el contrario recibe información de un producto que ya se encuentra en su base de datos lo que hace es actualizar la hora de la tabla correspondiente a los productos en stock, ya que si durante 15 minutos no ha recibido ningún mensaje sobre algún producto guardado en la base de datos éste se considera que ya no se encuentra en el almacén y es eliminado. Se ha considerado un periodo de 15 minutos, ya que los motes adheridos a cada producto envían un mensaje cada 5 minutos y se permite que se hayan producido un máximo de 3 errores al intentar enviar-recibir el mensaje.

Si en algún momento se desea detener la aplicación para que no procese ningún mensaje, habrá que clickar sobre el botón “Parar”, que se encuentra en la misma posición que el botón “Iniciar” (véase Fig. D.3).

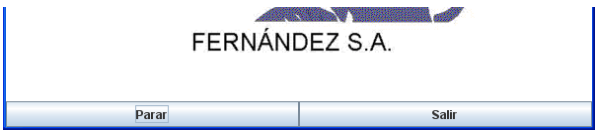

Fig. D.3 Botón “Parar”

D.3. Funcionalidades de los usuarios

En este punto se van a mostrar todas las funciones que tienen permitidos los administradores, los usuarios registrados y los usuarios sin registrar. Una vez clickado sobre el botón “Gestor” de la pantalla principal de la aplicación, aparecerá la pantalla que se muestra en la Fig. D.4.

Fig. D.4 Pantalla de Gestor de Datos

Esta pantalla se va a dividir en tres zonas según los permisos que se tengan.

- Zona 1: en esta zona se encuentra el botón “Administrador”, para acceder a ella es necesario estar registrado como administrador en la base de datos.

- Zona 2: en esta zona se encuentran los botones “Nuevo producto” y “Eliminar producto” desde los cuales se puede modificar la tabla donde se encuentran almacenados los productos en stock. Para acceder es necesario ser un usuario registrado.

- Zona 3: en esta zona se encuentran los botones “Ver productos” y “Actualizar”, mediante estos botones se actualiza la tabla que contiene los productos en stock y se muestra por pantalla. No es necesario estar registrado.

D.3.1. Administradores

Los administradores son los únicos que tienen privilegios para añadir, modificar o eliminar de la base de datos administradores, usuarios registrados, marcas y productos. Para poder acceder, la aplicación pide que se introduzca el *login* y el *password*, tal y como se muestra en la Fig. D.5.

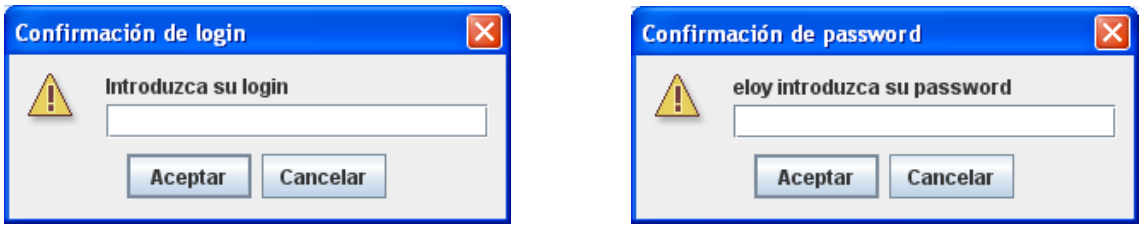

Fig. D.5 Confirmación de *login* y de *password*

Una vez confirmado que los datos introducidos son correctos, se accede a la pantalla desde la cual se puede realizar los privilegios del administrador nombrados anteriormente. En la Fig. D.6 se muestra la pantalla desde la cual el administrador puede acceder para realizar cambios.

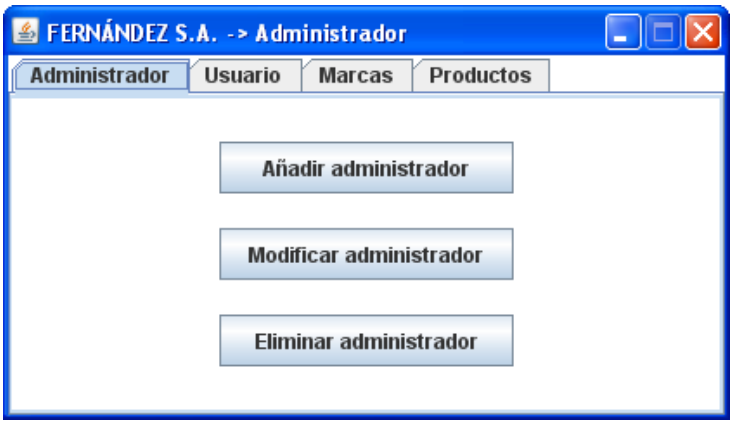

Fig. D.6 Pantalla del administrador, pestaña administrador

Como se puede observar en la figura anterior, la pantalla dispone de diferentes pestañas mediante las cuales se puede acceder para realizar cambios en la base de datos de los administradores como es este caso. Los usuarios registrados, las marcas y los productos, se verán posteriormente. Todas disponen de tres botones desde los cuales se pueden añadir, modificar o eliminar entradas de sus respectivas tablas de la base de datos. En las figuras siguientes se muestran los datos que hay que introducir según la acción que se desea realizar (véase Fig. D.7, Fig. D.8 y Fig. D.9).

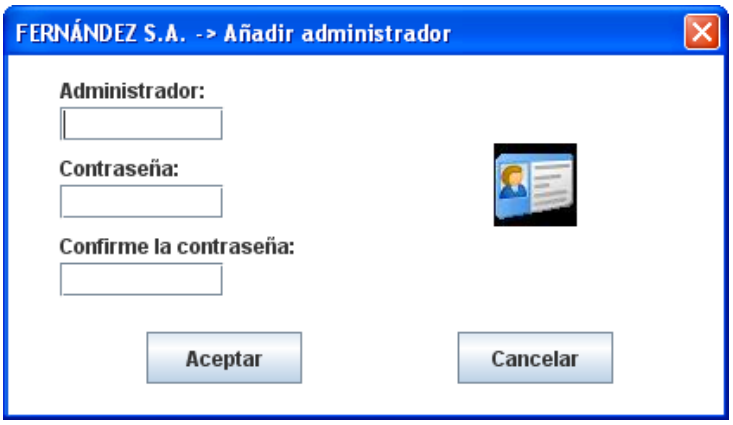

Fig. D.7 Añadir administrador

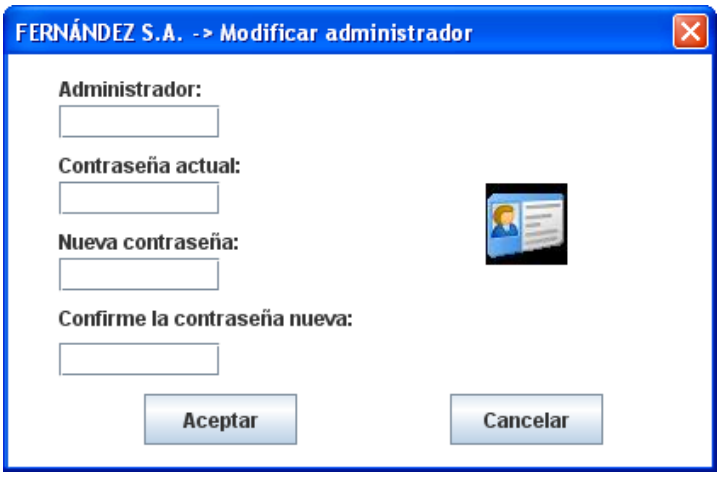

Fig. D.8 Modificar administrador

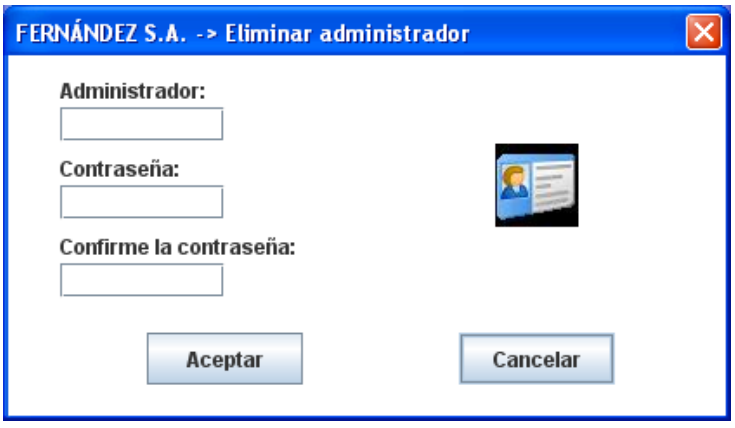

Fig. D.9 Eliminar administrador

En las figuras (Fig. D.10, Fig. D.11 y Fig. D.12) se mostrarán las pantallas cuando lo que se quiere modificar es un usuario.

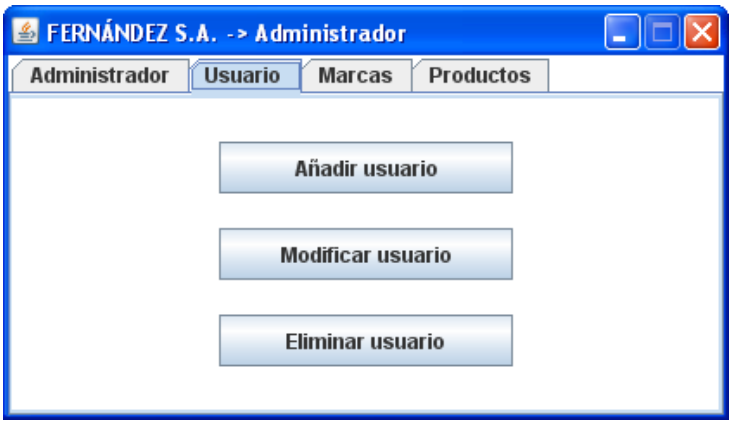

Fig. D.10 Pantalla del administrador, pestaña usuario

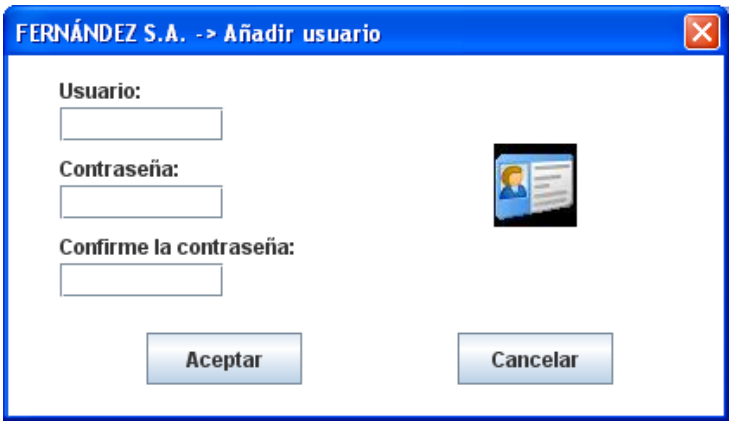

Fig. D.11 Añadir usuario

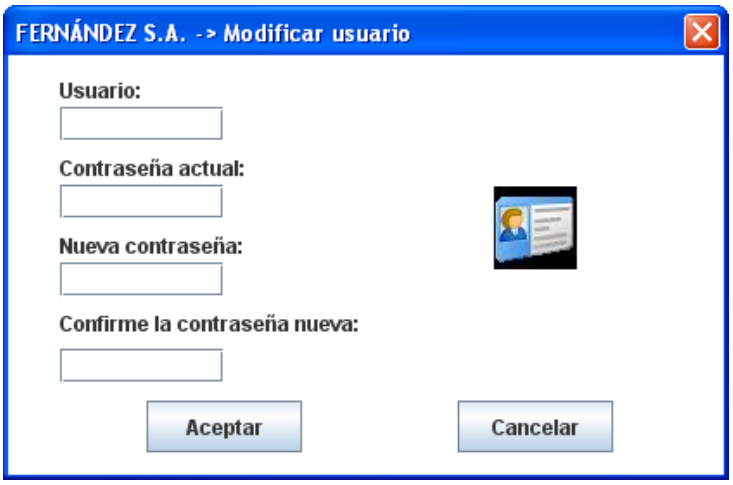

Fig. D.12 Modificar usuario

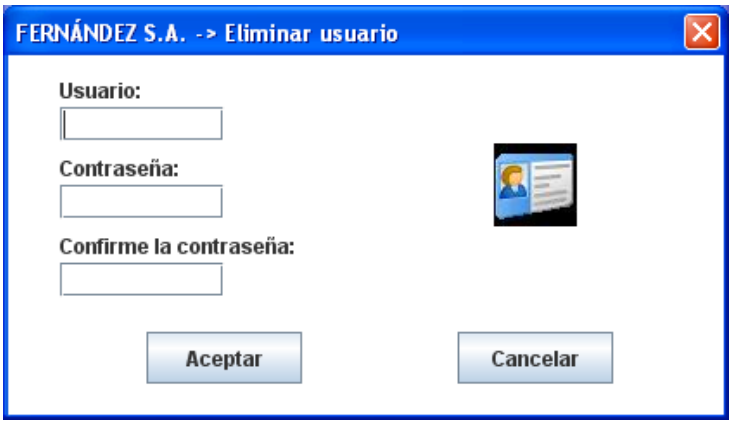

Fig. D.13 Eliminar usuario

Como se ha podido observar los datos que hay que rellenar para los usuarios son los mismos que para los administradores. A continuación se mostrarán los datos que hay que rellenar para el caso de marcas y productos, en estos casos en vez de nombres de usuarios y *passwords* habrá que introducir el nombre de la marca o producto y su respectivo código.

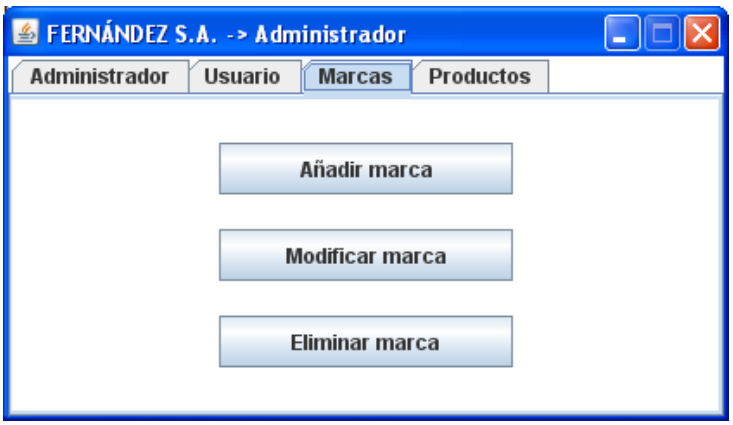

Fig. D.14 Pantalla del administrador, pestaña marcas

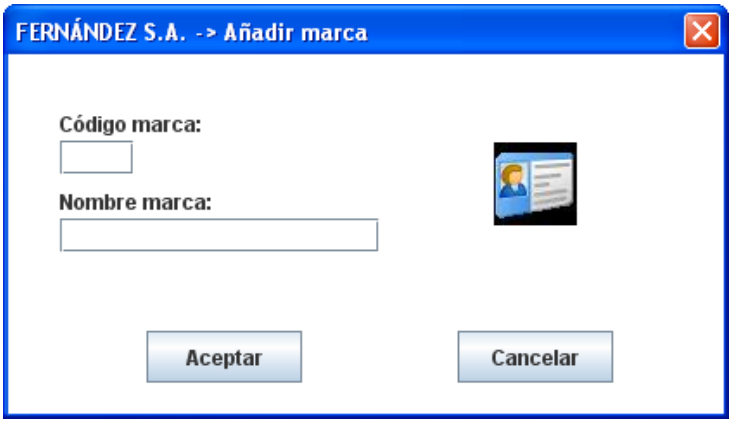

Fig. D.15 Añadir marca

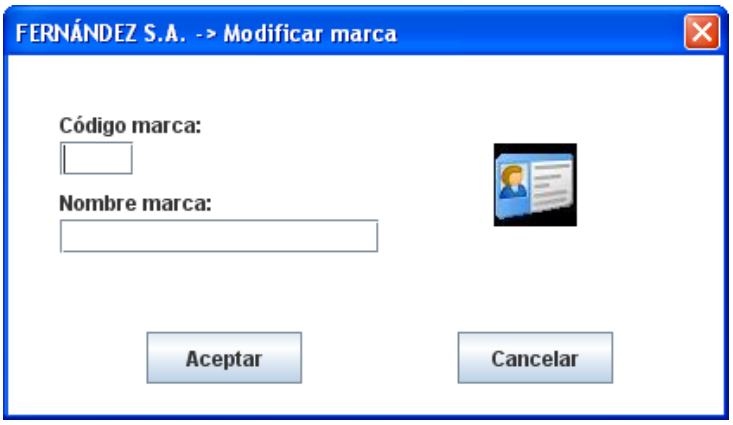

Fig. D.16 Modificar marca

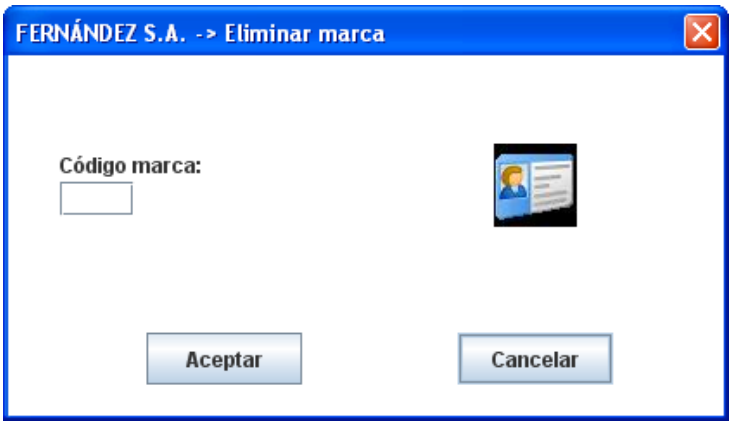

Fig. D.17 Eliminar marca

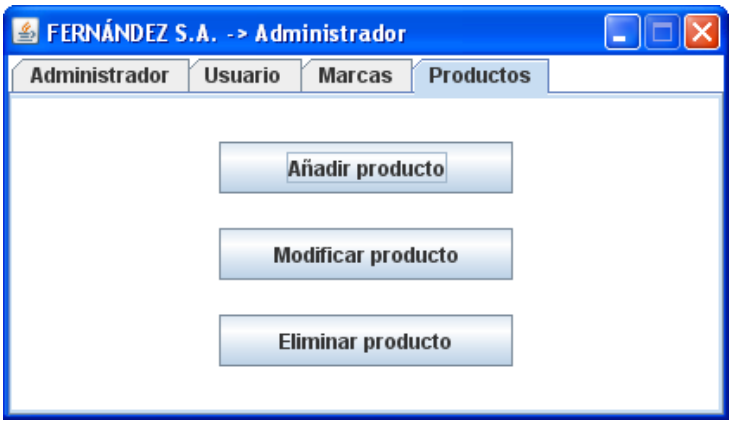

Fig. D.18 Pantalla del administrador, pestaña productos

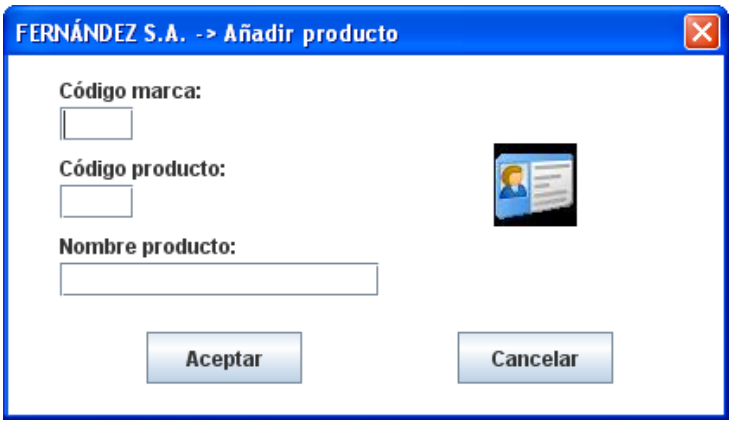

Fig. D.19 Añadir producto

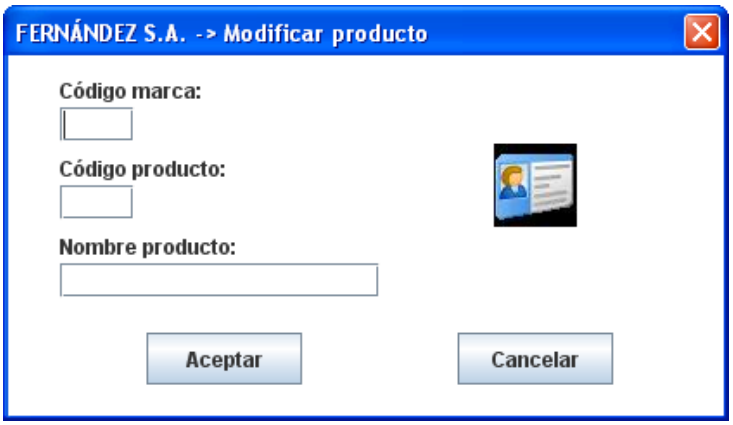

Fig. D.20 Modificar producto

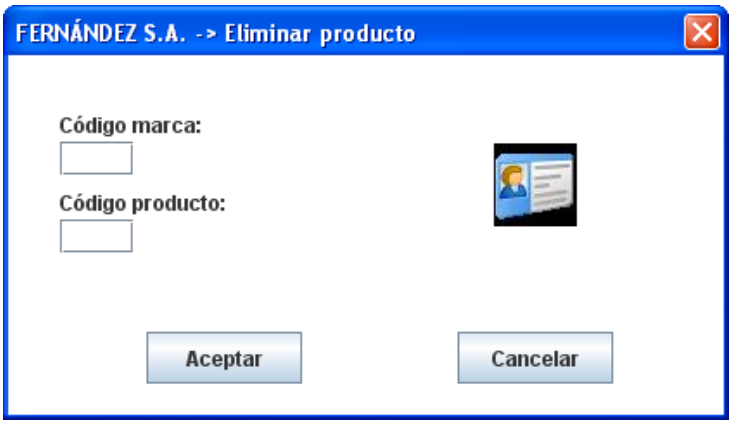

Fig. D.21 Eliminar producto

En el Anexo F, se nombrarán las diferentes pruebas realizadas para comprobar el correcto funcionamiento de todas estas funciones.

D.3.2. Usuarios registrados

Los usuarios registrados son los que tienen privilegios para modificar la tabla donde se almacenan los productos en stock. Éstos pueden añadir productos a esta tabla que por algún motivo ajeno a la aplicación no han sido detectados por ésta. En la Fig. D.22 se muestra la pantalla que aparece cuando se clicka sobre el botón “Nuevo producto”.

Fig. D.22 Pantalla nuevo producto

A la hora de añadir estos nuevos productos a la tabla donde se encuentran almacenados los productos en stock, el campo “código único” es rellenado por “00000” y en el campo “timer” se pone la hora actual del sistema del PC. Con la actualización automática que se hace cada 15 minutos o la manual, estos productos no son eliminados ya que también se realiza una comprobación del campo “código único” y al estar rellenados por “00000” sólo pueden ser eliminados de la base de datos a mano por un usuario registrado.

Para eliminar productos en stock de la base de datos hay que clickar sobre el botón “Eliminar producto” y aparecerá la pantalla que se muestra en la Fig. D.23.

Fig. D.23 Pantalla eliminar producto

A la hora de eliminar productos que han sido introducidos a mano y que su código úncio es “00000”, la aplicación eliminará de la base de datos el de mayor antigüedad.

D.3.3. Usuarios sin registrar

Los usuarios sin registrar sólo podrán actualizar la tabla de la base de datos donde se encuentran los productos en stock, a esto nos referimos cuando se decía actualizar la tabla manualmente. Para ello han de clickar sobre el botón actualizar y si se ha actualizado correctamente aparecerá por pantalla el mensaje que se muestra en la Fig. D.24.

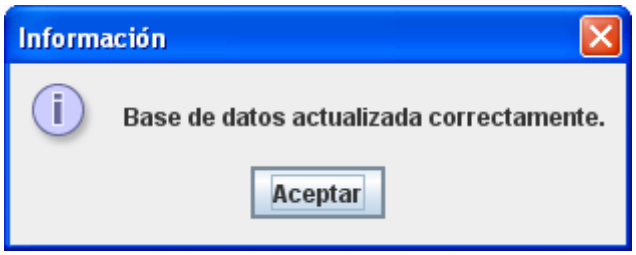

Fig. D.24 Mensaje de información de actualización de la base de datos.

La otra acción a la que tienen privilegio los usuarios sin registrar es la de poder ver por pantalla una tabla donde se muestran todos los productos en stock, véase Fig. D.25. Hay que añadir que antes de que la aplicación muestre por pantalla la tabla realiza una actualización de la tabla, igual que si se hubiera clickado sobre el botón “Actualizar”.

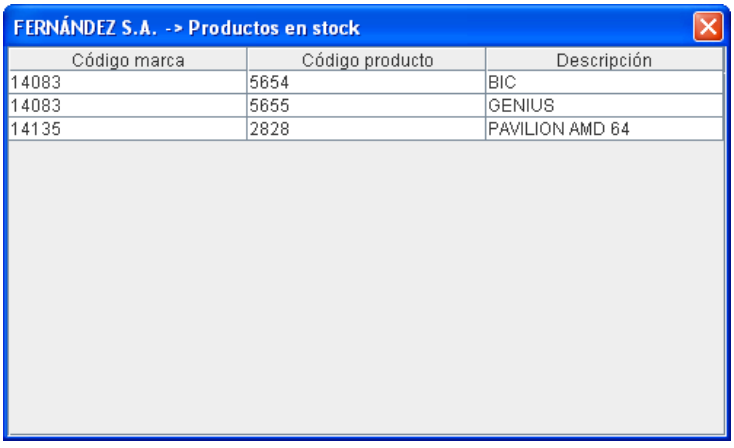

Fig. D.25 Tabla de los productos en stock

ANEXO E. GUÍA DE USUARIO DE LA APLICACIÓN MÓVIL

En este anexo se va a explicar como utilizar la aplicación móvil junto con todas las funciones que puede realizar esta aplicación. Todas las capturas que se verán a continuación han sido realizadas mediante las pruebas realizadas con el simulador Wireless Toolkit 2.5.2.

En la imagen de la Fig. E.1 se muestra la pantalla inicial de la aplicación, en ella se muestran los tres tipos de búsqueda de productos en stock que se pueden realizar: por marca, por código de producto o por fecha. Además en la parte inferior de la pantalla se puede observar el botón “Salir”, para salir de la aplicación.

Fig. E.1 Pantalla inicial de la aplicación móvil

Si se selecciona una búsqueda por marca, la pantalla que aparecerá es la que se muestra en la Fig. E.2, donde habrá que introducir el código de la marca que se esté buscando. También podemos observar dos botones en la parte inferior de la pantalla, el botón “Atrás” que te devuelve a la pantalla principal y el botón “Enviar” que es el encargado de enviar la petición siempre y cuando se haya introducido algún código de marca, en caso contrario, este botón no realizará ninguna petición.

Fig. E.2 Pantalla de búsqueda por marca

Una vez realizada la petición al servidor, pueden suceder dos cosas, por un lado que se hayan encontrado en la base de datos productos en stock pertenecientes a esta marca y se mostrará una tabla con los códigos de producto que están asignados a esta marca, el número de unidades de cada producto que hay actualmente en stock y los nombre de éstos, tal y como se muestra en la Fig. E.3. Si no se ha encontrado ningún producto se mostrará por la pantalla un mensaje indicándolo, como se puede ver en la Fig. E.3. Suceda una cosa u otra, la aplicación permitirá volver a acceder a la pantalla principal o salir del programa, mediante el uso de los botones “Inicio” y “Salir” respectivamente.

Fig. E.3 Pantalla de productos encontrados (imagen derecha) y pantalla de productos no encontrados (imagen izquierda)

Si se desea utilizar la búsqueda por productos, habrá que seleccionar esta búsqueda desde la pantalla principal de la aplicación. Una vez seleccionada, aparecerá por pantalla lo que se muestra en la Fig. E.4. Para este tipo de búsqueda hacen falta dos parámetros: el código de marca y el código de producto. Una vez introducidos los dos, el botón “Enviar” ya tendrá funcionalidad para poder realizar la petición.

Fig. E.4 Pantalla de búsqueda por producto

Lo que el usuario de la aplicación quiere conseguir con este tipo de búsqueda, es saber el número concreto de productos en stock que hay en el almacén de un determinado producto, para ello debe saber tanto en código de producto que busca como el código de la marca a la que pertenece. Si se han encontrado productos, la respuesta a la petición realizada es la que se muestra en la Fig. E.5, y si por el contrario no se han encontrado productos se muestra el mensaje que aparece en la figura E.5. Los botones “Inicio” y “Salir”, tiene la misma funcionalidad que en el caso de búsqueda por marca.

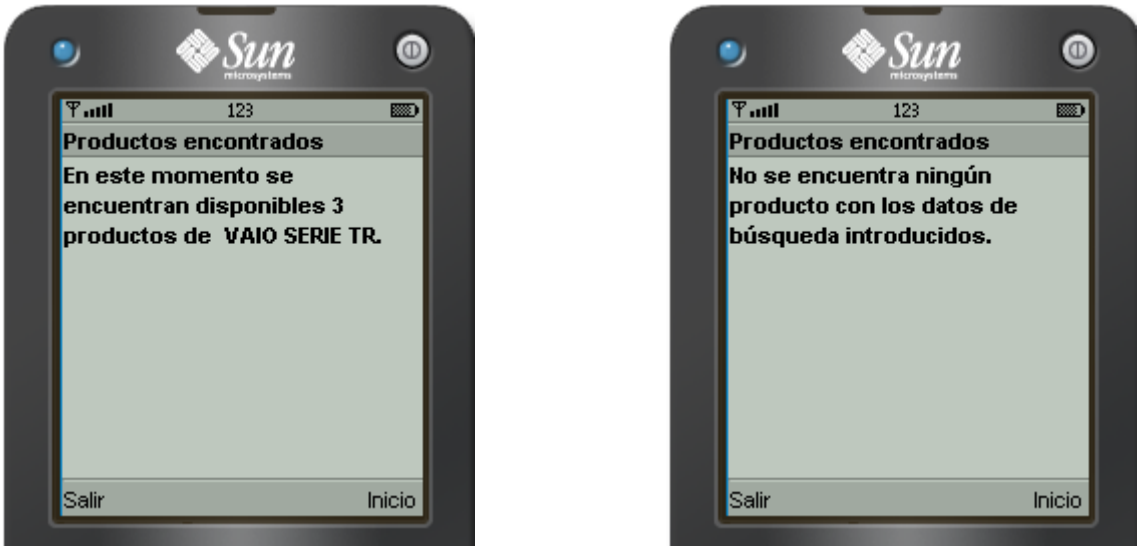

Fig. E.5 Pantalla de productos encontrados (imagen derecha) y pantalla de productos no encontrados (imagen izquierda)

Finalmente, el tercer tipo de búsqueda es por fecha. En este caso se ha dejado implementada una primera fase que llega hasta el envío de la petición, la cual ya no se realiza. Mediante este tipo de búsqueda lo que se quería conseguir es que si por ejemplo los motes también enviaban información relacionada por ejemplo, con la fecha de caducidad del producto o la fecha de fabricación de éste, etc., poder saber por ejemplo, el tiempo que les faltan a los productos para caducar o si ya hay alguno caducado. Para ello la aplicación permitía seleccionar un día cualquiera, por defecto sale la fecha actual pero desde ella se puede acceder a un calendario para cambiarla y después seleccionar uno o varios tipos de reglas de búsqueda, si sólo los productos con la fecha seleccionada, los anteriores a esta fecha o los posteriores, tal y como se muestra en la Fig. E.6.

Fig. E.6 Pantalla de búsqueda por fecha

Tal y como se ha dicho la implementación de esta búsqueda finaliza en el momento de enviar la petición, por ello aparecerá por pantalla el mensaje que se puede ver en la Fig. E.7. Desde esta pantalla igual que en los dos tipos de búsqueda anteriores se podrá volver a acceder a la pantalla principal o salir de la aplicación.

Fig. E.7 Pantalla del mensaje de información

ANEXO F. Control de pruebas de las funciones del administrador

A continuación se nombran todas las pruebas que se han realizado para la comprobación del correcto funcionamiento de todas las funciones del administrador en la aplicaciónPC.

Pestaña Administrador:

- AÑADIR
    - Comprobar que el administrador ya exista
    - Comprobar que la contraseña sea igual a la confirmación
    - Comprobar que se rellenan todos los campos
    - Comprobar que el usuario y el password tenga como máximo 10 caracteres
    - Comprobar que en usuario y en password sólo se pueda introducir código alfanumérico
    - Cuando el administrador ya existe borrar los datos del formulario para poderlos rellenar otra vez
    - Comprobar que lo inserta bien en la base de datos
    - Si lo ha insertado bien poner un mensaje indicándolo
- MODIFICAR
    - Comprobar que el administrador ya exista y la contraseña actual se corresponda
    - Comprobar que la contraseña nueva sea igual a la confirmación
    - Comprobar que se rellenan todos los campos
    - Comprobar que el usuario y el password tenga como máximo 10 caracteres
    - Comprobar que en usuario y en password sólo se pueda introducir código alfanumérico
    - Comprobar que lo modifica bien en la base de datos
    - Si lo ha modificado bien poner un mensaje indicándolo
- ELIMINAR
    - Comprobar que el administrador ya exista
    - Comprobar que la contraseña sea igual a la confirmación
    - Comprobar que al menos quede un administrador registrado en la base de datos
    - Comprobar que el usuario y el password tenga como máximo 10 caracteres
    - Comprobar que en usuario y en password sólo se pueda introducir código alfanumérico
    - Comprobar que se rellenan todos los campos

- Comprobar que lo elimina bien en la base de datos
- Si lo ha eliminado bien poner un mensaje indicándolo

Pestaña Usuario:

- AÑADIR
  - Comprobar que el administrador ya exista
  - Comprobar que la contraseña sea igual a la confirmación
  - Comprobar que se rellenan todos los campos
  - Comprobar que el usuario y el password tenga como máximo 10 caracteres
  - Comprobar que en usuario y en password sólo se pueda introducir código alfanumérico
  - Cuando el administrador ya existe borrar los datos del formulario para poderlos rellenar otra vez
  - Comprobar que lo inserta bien en la base de datos
  - Si lo ha insertado bien poner un mensaje indicándolo
- MODIFICAR
  - Comprobar que el administrador ya exista y la contraseña actual se corresponda
  - Comprobar que la contraseña nueva sea igual a la confirmación
  - Comprobar que se rellenan todos los campos
  - Comprobar que el usuario y el password tenga como máximo 10 caracteres
  - Comprobar que en usuario y en password sólo se pueda introducir código alfanumérico
  - Comprobar que lo modifica bien en la base de datos
  - Si lo ha modificado bien poner un mensaje indicándolo
- ELIMINAR
  - Comprobar que el administrador ya exista
  - Comprobar que la contraseña sea igual a la confirmación
  - Comprobar que se rellenan todos los campos
  - Comprobar que el usuario y el password tenga como máximo 10 caracteres
  - Comprobar que en usuario y en password sólo se pueda introducir código alfanumérico
  - Cuando el administrador ya existe borrar los datos del formulario para poderlos rellenar otra vez
  - Comprobar que lo elimina bien en la base de datos
  - Si lo ha eliminado bien poner un mensaje indicándolo

Pestaña marca:

- AÑADIR
    - Comprobar que el código de marca no exista
    - Comprobar que el código de marca tenga como máximo 5 caracteres
    - Comprobar que la descripción de la marca tenga como máximo 20 caracteres
    - Comprobar que se rellenan todos los campos
    - Comprobar que en código de marca sólo se puedan introducir números
    - Comprobar que en descripción sólo se pueda introducir código alfanumérico
    - Comprobar que lo inserta bien en la base de datos
    - Si lo ha insertado bien poner un mensaje indicándolo
- MODIFICAR
    - Comprobar que el código de marca ya exista
    - Comprobar que se rellenan todos los campos
    - Comprobar que el código de marca tenga como máximo 5 caracteres
    - Comprobar que la descripción de la marca tenga como máximo 20 caracteres
    - Comprobar que en código de marca sólo se puedan introducir números
    - Comprobar que en descripción sólo se pueda introducir código alfanumérico
    - Comprobar que lo modifica bien en la base de datos
    - Si lo ha modificado bien poner un mensaje indicándolo
- ELIMINAR
    - Comprobar que el código de marca no exista
    - Comprobar que se rellenan todos los campos
    - Comprobar que el código de marca tenga como máximo 5 caracteres
    - Comprobar que el código de marca tenga solo números
    - Comprobar que lo elimina bien en la base de datos
    - Si lo ha eliminado bien poner un mensaje indicándolo

Pestaña Producto:

- AÑADIR
    - Comprobar que el código de marca exista
    - Comprobar que el código de producto no exista
    - Comprobar que se rellenan todos los campos

  - Comprobar que el código de marca tenga como máximo 5 caracteres
  - Comprobar que en código de marca sólo se puedan introducir números
  - Comprobar que el código de producto tenga como máximo 5 caracteres
  - Comprobar que en código de marca sólo se puedan introducir números
  - Comprobar que la descripción del producto tenga como máximo 20 caracteres
  - Comprobar que en descripción sólo se pueda introducir código alfanumérico
  - Comprobar que lo inserta bien en la base de datos
  - Si lo ha insertado bien poner un mensaje indicándolo

- MODIFICAR

  - Comprobar que el código de marca se corresponda con el código de producto que se quiere modificar
  - Comprobar que se rellenan todos los campos
  - Comprobar que el código de marca tenga como máximo 5 caracteres
  - Comprobar que en código de marca sólo se puedan introducir números
  - Comprobar que el código de producto tenga como máximo 5 caracteres
  - Comprobar que en código de producto sólo se puedan introducir números
  - Comprobar que la descripción del producto tenga como máximo 20 caracteres
  - Comprobar que en descripción sólo se pueda introducir código alfanumérico
  - Comprobar que lo modifica bien en la base de datos
  - Si lo ha modificado bien poner un mensaje indicándolo

- ELIMINAR

  - Comprobar que el código de marca se corresponda con el código de producto que se quiere eliminar
  - Comprobar que se rellenan todos los campos
  - Comprobar que el código de marca tenga como máximo 5 caracteres
  - Comprobar que en código de marca sólo se puedan introducir números
  - Comprobar que el código de producto tenga como máximo 5 caracteres
  - Comprobar que en código de producto sólo se puedan introducir números
  - Comprobar que la descripción del producto tenga como máximo 20 caracteres

- Comprobar que en descripción sólo se pueda introducir código alfanumérico
- Comprobar que lo elimina bien en la base de datos
- Si lo ha eliminado bien poner un mensaje indicándolo